6-2020

An extended framework of privacy-preserving computation with flexible access control

Wenxiu DING

Rui HU

Zheng YAN

Xinren QIAN

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

See next page for additional authors

Citation

1

Author
Wenxiu DING, Rui HU, Zheng YAN, Xinren QIAN, Robert H. DENG, Laurence T. YANG, and Mianxiong DONG

Published in IEEE Transactions on Network and Service Management, 2020 June, 17 (2), 918-930
https://doi.org/10.1109/TNSM.2019.2952462

An Extended Framework of Privacy-Preserving Computation With Flexible Access Control

Wenxiu Ding, *Member, IEEE*, Rui Hu, Zheng Yan, *Senior Member, IEEE*, Xinren Qian, Robert H. Deng, *Fellow, IEEE*, Laurence T. Yang, *Senior Member, IEEE*, and Mianxiong Dong, *Member, IEEE*

***Abstract*—Cloud computing offers various services based on outsourced data by utilizing its huge volume of resources and great computation capability. However, it also makes users lose full control over their data. To avoid the leakage of user data privacy, encrypted data are preferred to be uploaded and stored in the cloud, which unfortunately complicates data analysis and access control. In particular, few existing works consider the fine-grained access control over the computational results from ciphertexts. Though our previous work proposed a framework to support several basic computations (such as addition, multiplication and comparison) with flexible access control, privacy-preserving division calculations over encrypted data, as a crucial operation in many statistical processes and machine learning algorithms, is neglected. In this paper, we propose four privacy-preserving division computation schemes with flexible access control to fill this gap, which can adapt to various application scenarios. Furthermore, we extend a division scheme over encrypted integers to support privacy-preserving division over multiple data types including fixed-point numbers and fractional numbers. Finally, we give their security proof and show their efficiency and superiority through comprehensive simulations and comparisons with existing work.**

***Index Terms*—Cloud computing, secure division computation, privacy preservation, access control, data security.**

Manuscript received May 15, 2019; revised August 10, 2019, October 25, 2019, and November 1, 2019; accepted November 4, 2019. Date of publication November 8, 2019; date of current version June 10, 2020. This work is sponsored by National Postdoctoral Program for Innovative Talents (grant BX20180238), the NSFC (grants 61672410, 61802293 and U1536202), the Project funded by China Postdoctoral Science Foundation (grant 2018M633461), the Academy of Finland (grants 308087 and 314203), the Key Lab of Information Network Security, Ministry of Public Security (grant C18614), the Shaanxi innovation team project (grant 2018TD-007), the 111 project (grants B16037 and B08038), the Fundamental Research Funds for the Central Universities (grant JB191504). The associate editor coordinating the review of this article and approving it for publication was S. Scott-Hayward. *(Corresponding author: Zheng Yan.)*

W. Ding, R. Hu, and X. Qian are with the School of Cyber Engineering, Xidian University, Xi'an 710126, China (e-mail: wxding@xidian.edu.cn; ruihu2019@126.com; xinrenqian@gmail.com).

Z. Yan is with the State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Department of Communications and Networking, Aalto University, 00076 Espoo, Finland (e-mail: zheng.yan@aalto.fi).

R. H. Deng is with the School of Information Systems, Singapore Management University, Singapore (e-mail: robertdeng@smu.edu.sg).

L. T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: ltyang@stfx.ca).

M. Dong is with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran 050-8585, Japan (e-mail: mx.dong@csse.muroran-it.ac.jp).

I. Introduction

CLOUD computing can efficiently store and process data in the Internet by taking advantage of its huge volume of resources and great computation. The arrival of cyberization era has led to the demand of massive data generation, analysis and processing, which causes high computation overhead that cannot be handled by local devices [2], [3], [4]. Outsourcing computing to the cloud can greatly benefit resource-constrained users [5]. However, the dynamic, random and open nature of cloud computing makes it hard to be fully trusted. It may disclose user private data, which seriously impacts user privacy and threatens data security. Therefore, cloud users prefer to first encrypt their sensitive data and then outsource ciphertext to the cloud. However, encryption introduces new challenges for data analysis and sharing as described below.

First, encryption complicates data processing and analysis, especially for division. Though partial/fully homomorphic encryption (PHE/FHE) can be applied to perform operations over encrypted data, PHE algorithms can only support multiplication and addition over encrypted data [6], [7]. FHE algorithms can realize division computation over encrypted data but introduce high computational and communication overhead [8], [9], thus they are not applicable and efficient in practice.

Second, flexible access control over outsourced data computation results is still an open issue. Most existing homomorphic encryption systems only support single-user access to the results [10]. The past literatures mainly focus on the access control over the outsourced data, but ignore the security requirement of access control over the processing results [11], which is however especially significant and essential to support various intelligent applications, such as smart grid, smart transportation, health-care services, and so on.

In our previous work, a framework [1] with two non-colluding servers was proposed to achieve encrypted data computation with flexible access control over the computation results in a privacy-preserving manner. However, division computation is missed therein owing to its complexity. In order to enhance the applicability, flexibility and scalability of our framework, we extend and complement it by providing privacy-preserving division computation on the basis of our previous system model.

In this paper, we propose several novel schemes to realize privacy-preserving division over ciphertext with fine-grained access control over division results. Specifically, this work has the following contributions:

- We first extend our previous framework to support division computation over encrypted integers without any transformations or decompositions. Our scheme can provide the ciphertext of quotient and remainder values from outsourced encrypted data to specified data requester or user group, which preserve the confidentiality of both original data and final computation result.
- We realize flexible access control over the division results in a privacy-preserving manner. Our scheme can adapt to different application scenarios with sound scalability.
- We further extend our scheme in terms of integers to support privacy-preserving division computation over other types of data including fractional numbers and fixed-point numbers, which obviously improves the precision and practicability of division computation over ciphertext.
- We prove the correctness and security of our proposed schemes, and further demonstrate its efficiency and scalability through extensive simulations and comparisons with existing works.

The rest of this paper is organized as follows. In Section II, we briefly overview the existing work on secure division computation and analyze their pros and cons, followed by the system model, attack model and design goals of our schemes in Section III. Section IV presents our proposed four division schemes over integers as well as their extension for supporting fixed point numbers and fractional numbers. Then we give correctness proof, security analysis and performance evaluation in Section V. Finally, we conclude the whole paper in the last section.

II. Related Work

Secure division computation plays a crucial role in secure statistical analysis [12], [13], secure clustering in machine learning [14] and secure recommender systems [15]. Few researchers implement secure division over encrypted data based on FHE due to its high complexity. And most of related works apply PHE, which mainly falls into two categories. One is based on arithmetic transformations to convert the secure division computation into addition and multiplication over encrypted data, and the other achieves secure division based on a secure bit decomposition protocol. However, both methods suffer from either high communication overhead or high computational complexity. In addition, they cannot provide flexible access control over division result to support multiple result requestors.

A. Secure Division Based on Arithmetic Transformations

Franz *et al.* [16] chose a tuple $(\rho_x, \sigma_x, \tau_x)$ to represent a value χ which belongs to a certain interval $[-l;\ +l]$ with $l > 0$, where ρ_x is a nonzero flag, σ_x encodes the sign of the value x and τ_x indicates the absolute of the value. Then the division result can be computed by basic operations on corresponding element through function $LDIV([\overline{x}], [\overline{y}])$. Though the representation of numbers can support secure computations on non-integers, its final computational division result is an approximation with bounded relative error and encoding increases the overhead of data preprocessing.

To overcome this issue and get an accurate result, Dahl *et al.* [17] performed a Taylor expansion on the reciprocal of a denominator to transform the division computation over encrypted data into multiplication and addition over encrypted data. Though the proposed protocol can guarantee the privacy of division, the implementation of several sub-protocols bring high computational overhead. In addition, the frequent interactions between two servers bring high communication overhead.

Veugen [18] presented three protocols for dividing encrypted data based on a client-server model in which the ciphertext $[x]$ and its corresponding decryption key K are held by the client and the server, respectively. But the divisor is known to the server in these three protocols, which unfortunately cannot ideally support privacy preservation.

In order to improve the precision of division results, Catrina and Saxena [19] tried to approximately get a division result over two floating point numbers by applying the Goldschmidt's method [20]. But this scheme cannot support division computation over encrypted input data. To overcome this weakness, Ugwuoke *et al.* [21] designed a division protocol to support encrypted floating point numbers based on homomorphic encryption. However, both of the above two division schemes use fixed rounds of iterative computations to guarantee fixed precise of results, which results in high computational overhead.

B. Secure Division Based on Bit Decomposition Protocol

The modulo value operation limits the length of the data in division computation. To guarantee the confidentiality of both the divisor and the dividend, one way is to add some random numbers into these values. But this makes it difficult to gain an accurate quotient from the masked data. Hence, some studies use the secure bit-decomposition (SBD) protocol [22] to realize secure division [23], [24]. After data providers upload their encrypted data, the cloud first decomposes encrypted data as binary string and then executes division to get a quotient and a remainder by operating secure bit shift. But the bit decomposition protocol is generally very complicated, thus hard to be deployed.

Notably, all aforementioned works ignore the access control over the division results from ciphertext. In our previous work [1], flexible access control over seven basic operations (such as addition, subtraction, etc.) from ciphertext were achieved based on key-policy attributed-based encryption (KP-ABE) [25], [26], [27], [28]. However, division computation over ciphertext is still not supported and needs further investigation.

III. Problem Statements

A. System Model

The division function is a novel addition and complement to our previous work, which makes our previous framework

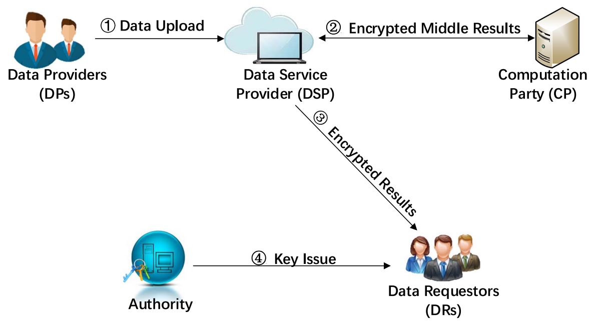

Fig. 1. System model.

of privacy-preserving computations more generic and applicable. In this paper, we follow the same models as in [1] for the purpose of extending its functionality to further support division computation. Concretely, the system is composed of five kinds of entities in Fig. 1.

1) Data Service Provider (DSP) provided by a cloud server takes the responsibility of data storage and computation service.
2) Computation Party (CP) can be a private cloud service provider or a department in charge, which mainly offers the service of data computation and access control.
3) Data Providers (DPs) are cloud service consumers that collect or generate data and upload them to DSP for efficient storage and computation.
4) Data Requesters (DRs) as data consumers request for the processing results. A DR can be a DP.
5) Authority is fully trusted and in charge of key management.

B. Attack Model

In the system model above, we assume all entities except the authority are semi-trusted. The authority acts honestly and has no collusion with any other entities, while other entities strictly follow the design of system protocols but are still curious about others' data. In addition, we assume that the DSP and the CP would not collude with each other because their collusion will decrease their reputations and impact their individual interests. Herein, an adversary $\mathcal{A}^*$ is introduced to the attack model. Its goal is to gain the raw data by challenging data users (either a DR or a DP) with some special capabilities as follows.

1) $\mathcal{A}^*$ can eavesdrop all communication channels except the ones between Authority and users to get those transmitted messages;
2) $\mathcal{A}^*$ may compromise one server (either DSP or CP) to guess the raw data through the ciphertexts transmitted between themselves and other users;
3) $\mathcal{A}^*$ may compromise one server (either DSP or CP) and some DPs to guess the final processing results;
4) $\mathcal{A}^*$ may also compromise one server (either DSP or CP) and the DR to guess the original data provided by DPs.

The adversary $\mathcal{A}^*$ may compromise all entities, but it cannot compromise the DSP and the CP simultaneously and cannot compromise the challenged DR or DP.

TABLE I
NOTATION DESCRIPTION

Symbols	Description
g_0	The system generator in ABE
g	The system generator in HRES
n	The system parameter
(sk_i, pk_i)	The public/private key pair of entity i
$PK = pk_{DSP}^{sk_{CP}} = pk_{CP}^{sk_{DSP}}$	The public key generated through the keys of DSP and CP
m_i	The original data provided by DP i
$[m]$	The ciphertext of data m under PK
$[m]_{pk_i}$	The ciphertext of m under pk_i
$\lfloor \cdot \rfloor$	The quotient of division operation
Q	The quotient of division operation
R	The remainder of division operation
$L(\cdot)$	The length of a piece of data

C. Design Goals

1) Confidentiality: Our schemes should guarantee that only the authorized entities can access the final computation results and that no entity can access the raw data provided by the data providers.

2) Correctness: The proposed schemes can offer DRs accurate division results.

IV. Privacy-Preserving Division Schemes With Flexible Access Control

A. Notations and Preliminaries

1) Notations: In order to get a better understanding of scheme details, Table I lists some key notations used throughout this paper.

2) Ciphertext-Policy Attribute-Based Encryption (CP-ABE): Herein, we introduce CP-ABE for applying it to support fine-grained access control, which guarantees the security of raw data through access policy. Generally, we mainly use the following four algorithms of CP-ABE in [29]:

$Setup^{ABE} \rightarrow (PK', MSK')$: This algorithm first selects a bilinear group G_0 of prime order p with generator g_0 as well as two random exponents $\alpha, \beta \in Z_p$. Then, it outputs the public key PK' and a master secret key MSK'.

$$PK' = \left(G_0, g_0, h = {g_0}^{\beta}, f = {g_0}^{1/\beta}, e(g_0, g_0)^{\alpha}\right) \quad (1)$$
$$MSK' = (\beta, {g_0}^{\alpha}). \quad (2)$$

$Enc^{ABE}(M, \mathcal{T}, PK') \rightarrow CK'$: The algorithm inputs message M, access policy $\mathcal{T}$ and PK'. It chooses a polynomial q_x for each node and sets $q_x(0)$ according to Eq. (3) except for the root node x. Then it outputs CK' as shown in Eq. (4), where s is a randomly chosen number and Y is the set of leaf nodes in $\mathcal{T}$.

$$q_x(0) = q_{parent(x)}(index(x)) \quad (3)$$
$$CK' = \Big(\mathcal{T}, \tilde{C} = Me(g_0, g_0)^{\alpha s}, C = h^s, \forall y \in Y : C_y = {g_0}^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\Big). \quad (4)$$

$KeyGen^{ABE}(MSK', \mathcal{S}) \rightarrow SK'$: The key generation algorithm takes in *MSK′* and a set of attributes $\mathcal{S}$, and generates a secret key SK'.

$$SK' = \Big(D = {g_0}^{(\alpha+r)/\beta}, \\ \forall j \epsilon \mathcal{S} : D_j = {g_0}^{r} \cdot H(j)^{r_j}, D_j' = {g_0}^{r_j}\Big). \quad (5)$$

$Dec^{ABE}(PK', SK', CK') \rightarrow M$: If the set of attributes embedded in the private key *SK′* satisfy the access policy, the decryption algorithm can successfully decrypt the ciphertext *CK′* to get the message *M*.

Notably, KP-ABE [25] can also be used in our scheme. If two pieces of data are encrypted with the same access policy, the CP-ABE is multiplicative homomorphic. That is, given CP-ABE ciphertext of M_1 and M_2 encrypted with the same access structure, the ciphertext of $M_1 * M_2$ can be obtained through the multiplication of these two pieces of ciphertext with Eq. (6), denoted as HE^{ABE}. We prove the homomorphism of CP-ABE later.

$$Enc^{ABE}\big(M_1 * M_2, \mathcal{T}, PK'\big) \\ = Enc^{ABE}\big(M_1, \mathcal{T}, PK'\big) * Enc^{ABE}\big(M_2, \mathcal{T}, PK'\big) \quad (6)$$

In this paper, we introduce the CP-ABE in [29] for access control, which only considers the static scenarios. Hence, our scheme cannot support dynamic environment and fails to support user revocation. In order to solve this issue, we can apply a revocable ABE [30] and integrate into our scheme or use a blacklist to block revoked users. User revocation is not the focus of this paper and can play as a part of our future work.

3) Homomorphic Re-Encryption System (HRES): We proposed homomorphic re-encryption system (HRES) in [31], which lays the foundation of all seven data operations in our previous work [1]. Herein, we also apply HRES for data outsourcing in this paper. A brief introduction to HRES is presented as below.

Key Generation (KeyGen): During system setup, it generates the public parameters: a big integer *n*, a system generator *g*. In addition, each entity *i* (including DSP and CP) generates one key pair (sk_i, pk_i). Further DSP and CP negotiate their Diffie-Hellman key *PK*. which should be issued to its customers. Hence, the public system parameters include {*g, n, P K*}.

Encryption: $Enc(m, pk_i) \rightarrow [m]_{pk_i}$, as shown in Eq. (7).

$$[m]_{pk_i} = \{(1 + m * n)pk_i^r, g^r\} \bmod n^2 \quad (7)$$

Decryption: $Dec([m]_{pk_i}, sk_i) \rightarrow m$.

Encryption With PK: $EncTK(m, PK) \rightarrow [m]_{PK}$. The ciphertext under *PK*, will be denoted as [*m*] in the following sections.

Partial Decryption With sk_{DSP}: $PDec1([m], sk_{DSP}) \rightarrow [m]_{pk_{CP}}$.

Partial Decryption With sk_{CP}: $PDec2([m]_{pk_{CP}}, sk_{CP}) \rightarrow m$.

Besides the additive homomorphism shown in Eq. (8), Eqs. (9), (10) and (11) illustrate several features of HRES, where *r* in Eq. (9) denotes a random number and $([m]_{pk_i})^{1,t}$ in Eq. (11) represents that an exponential operation that only performs on the first part of the ciphertext.

$$[m_1]_{pk_i} * [m_2]_{pk_i} = [m_1 + m_2]_{pk_i} \quad (8)$$

$$[r * m]_{pk_i} = \Big([m]_{pk_i}\Big)^r \quad (9)$$

$$\Big([m]_{pk_i}\Big)^{n-1} = [-m]_{pk_i} \quad (10)$$

$$\Big([m]_{pk_i}\Big)^{1,t} = \Big\{\{(1 + m * n)pk_i^r\}^t, g^r\Big\} \bmod n^2. \quad (11)$$

B. Privacy-Preserving Division Over Integers With Flexible Access Control

In this section, we design four different division schemes to support division computation over encrypted integers, which are suitable for different scenarios. First of all, we outline four schemes.

The first scheme aims to obtain the quotient value from two encrypted data that can be accessed by a specified data requester DR. Given two piece of encrypted data $[m_1]$ and $[m_2]$, it can provide the ciphertext of division result $[\lfloor m_1/m_2 \rfloor]pk_{DR}$, which guarantees that only the targeted data requester can access the quotient.

The second scheme is designed to enable flexible access control over computational result. Given ciphertext $[m_1]$ and $[m_2]$, the second scheme can provide the division result $[\lfloor m_1/m_2 \rfloor]_{pk_{ck}}$, while the corresponding secret key is encrypted via ABE. Hence, the data requesters who satisfy the access policies can obtain the secret key and get the final quotient.

The third scheme further calculates division remainder compared to the first scheme to provide an accurate division computational result.

Similarly, the fourth scheme is proposed to further provide the remainder of division based on the second scheme.

In what follows, we introduce the details of four schemes.

1) Privacy-Preserving Division Computations for a Targeted Data Requester (Scheme 1): In Scheme 1, DSP and CP process data collected from DP by following the procedure as shown in Fig. 2.

Step 1 (System Setup @ All Entities): The authority invokes *KeyGen* to complete the setup of HRES.

Step 2 (Data Upload @ DPs): DPs call $EncTK(m_i, PK)$ to encrypt their data m_i as $[m_i]$ and then upload them to DSP. To avoid the overflow of middle results, it should ensure $L(m_1) < 3L(n)/4$ and $L(m_2) < L(n)/2$.

Step 3 (Data Preparation @ DSP): DSP first randomly selects two numbers r_1, r_2 and conceals the raw data to get $[m_1 r_1]$, $[m_2 r_1]$ and $[m_2 r_1 r_2]$ through Eq. (8), where $L(r_i) < L(n)/4$. Then it acquires $[m_1 r_1 + m_2 r_1 r_2]$ by using Eq. (12). Next, DSP performs partial decrytion on $[m_2 r_1]$ and $[m_1 r_1 + m_2 r_1 r_2]$ to get $[m_2 r_1]_{pk_{CP}}$ and $[m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}}$ by calling $PDec1(*, sk_{DSP})$. Afterwards, DSP sends the data packet $([m_2 r_1]_{pk_{CP}}, [m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}})$ to CP.

$$[m_1 r_1 + m_2 r_1 r_2] = [m_1 r_1] * [m_2 r_1 r_2]. \quad (12)$$

Step 4 (Data Process @ CP): CP calls $PDec2(*, sk_{CP})$ to decrypt the data received from DSP and get the masked

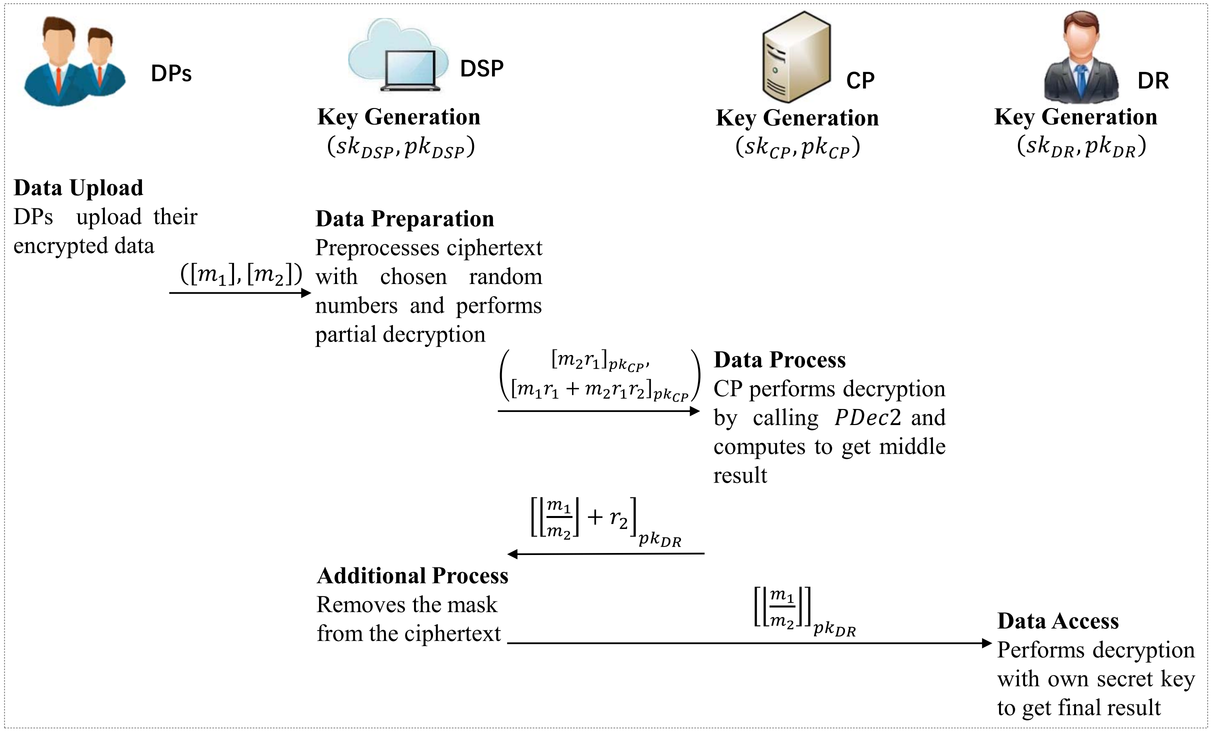
DPs
DSP
CP
DR
Key Generation
(sk_{DSP}, pk_{DSP})
Key Generation
(sk_{CP}, pk_{CP})
Key Generation
(sk_{DR}, pk_{DR})
Data Upload
DPs upload their encrypted data
$([m_1], [m_2])$
Data Preparation
Preprocesses ciphertext with chosen random numbers and performs partial decryption
$\left([m_2 r_1]_{pk_{CP}}, [m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}}\right)$
Data Process
CP performs decryption by calling $PDec2$ and computes to get middle result
$\left[\left\lfloor\frac{m_1}{m_2}\right\rfloor + r_2\right]_{pk_{DR}}$
Additional Process
Removes the mask from the ciphertext
$\left[\left\lfloor\frac{m_1}{m_2}\right\rfloor\right]_{pk_{DR}}$
Data Access
Performs decryption with own secret key to get final result

Fig. 2. The procedure of division computation for a targeted data requester.

data $m_2 r_1$ and $(m_1 r_1 + m_2 r_1 r_2)$. Then CP performs division on plaintext according to Eq. (13), where $\lfloor \frac{m_1}{m_2} \rfloor$ represents the quotient and ignores the remainder. CP further calls $Enc(*, pk_{DR})$ to encrypt the computational result and sends the ciphertext $[\lfloor \frac{m_1}{m_2} \rfloor + r_2]_{pk_{DR}}$ to DSP.

$$(m_1 r_1 + m_2 r_1 r_2)/m_2 r_1 = \left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2. \tag{13}$$

Step 5 (Additional Process @ DSP): DSP encrypts the random number r_2 as $[r_2]_{pk_{DR}}$ and computes $([r_2]_{pk_{DR}})^{n-1}$. Then DSP removes the mask from received ciphertext with Eq. (14).

$$\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{DR}} * \left([r_2]_{pk_{DR}}\right)^{n-1} = \left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{DR}} \tag{14}$$

Note: Herein, we use $\lfloor \frac{m_1}{m_2} \rfloor$ to represent the quotient regardless of remainder.

Step 6 (Data Access @ DR): Upon receiving final ciphertext from DSP, the targeted DR can call $Dec([\lfloor \frac{m_1}{m_2} \rfloor]_{pk_{DR}}, sk_{DR})$ to get the final quotient of division.

2) Privacy-Preserving Division Computations With Flexible Access Control (Scheme 2): In Scheme 2, DSP and CP process encrypted data from DP as shown in Fig. 3.

Step 1 (System Setup @ All Entities): The authority first invokes *KeyGen* and $Setup^{ABE}$ to set up the system and generate the key parameters PK' and MSK' of ABE algorithm. Then, the authority publishes public parameters to its service consumers.

Step 2 (Data Upload @ DPs): Similar to Step 2 of Scheme 1, DPs upload encrypted data $[m_i]$ to DSP.

Step 3 (Data Preparation @ DSP): DSP selects two random numbers $r_1, r_2 \in [1, n/4]$, and preprocesses data to mask raw data, which is same as Scheme 1. Similarly, DSP sends the data packet $([m_2 r_1]_{pk_{CP}}, [m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}})$ to CP.

Step 4 (Data Process @ CP): CP calls $PDec2(*, sk_{CP})$ to decrypt the received data from DSP and performs division on masked plaintext with Eq. (13). Then it encrypts the computational result by calling $Enc(*, pk_{CP})$ and sends encrypted data $[\lfloor \frac{m_1}{m_2} \rfloor + r_2]_{pk_{CP}}$ to DSP.

Step 5 (Data Reprocess @ DSP): DSP chooses a partial key ck_1 and sets a random number c_1 as $(ck_1)^{-1} \bmod n$. Furthermore, it removes the mask from received ciphertext through Eq. (15) and then uses Eq. (16) to perform exponential compuation. Finally DSP sends $[c_1 \lfloor \frac{m_1}{m_2} \rfloor]_{pk_{CP}}$ to CP.

$$\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{CP}} * \left([r_2]_{pk_{CP}}\right)^{n-1} = \left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}} \tag{15}$$

$$\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}} = \left(\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}}\right)^{c_1}. \tag{16}$$

Step 6 (Data Reprocess @ CP): CP first calls $PDec2(*, sk_{CP})$ to decrypt received ciphertext as $c_1 \lfloor \frac{m_1}{m_2} \rfloor$. Then it chooses a partial key ck_2 to generate a key pair (ck_2, pk_{ck_2}) and calls $Enc(*, pk_{ck_2})$ to encrypt the data. In addition, CP calls $Enc^{ABE}(ck_2, \mathcal{T}, PK')$ to obtain the ABE ciphertext CK_2, which is sent to DSP along with the ciphertext $[c_1 \lfloor \frac{m_1}{m_2} \rfloor]_{pk_{ck_2}}$.

Step 7 (Additional Process @ DSP): DSP operates partial modular computation on received ciphertext with its partial key ck_1 according to Eq. (11) and gets the ciphertext $[\lfloor \frac{m_1}{m_2} \rfloor]_{pk_{ck}}$ with Eq. (17). Then it calls $Enc^{ABE}(ck_1, \mathcal{T}, PK')$ to get CK_1 and obtains an encrypted access key CK through

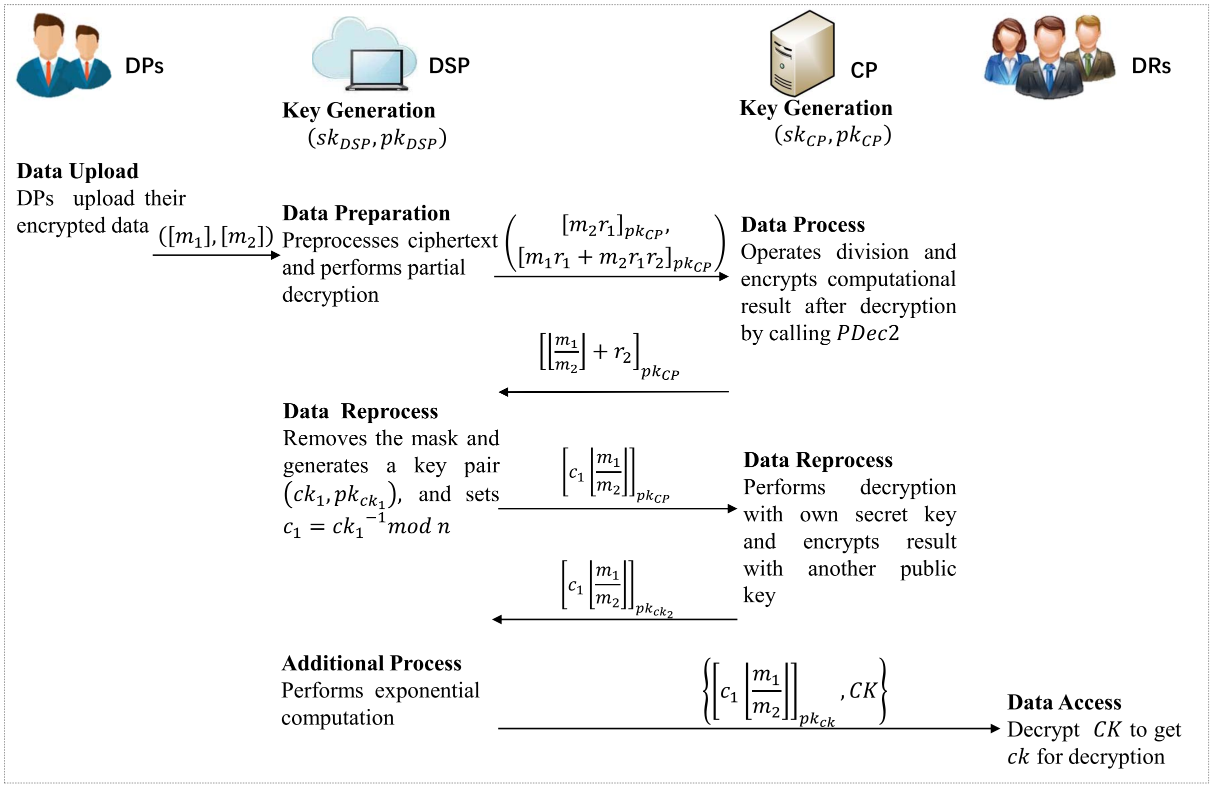
DPs
DSP
CP
DRs
Key Generation (sk_{DSP}, pk_{DSP})
Key Generation (sk_{CP}, pk_{CP})
Data Upload
DPs upload their encrypted data
$([m_1], [m_2])$
Data Preparation
Preprocesses ciphertext and performs partial decryption
$\begin{pmatrix}[m_2 r_1]_{pk_{CP}}, \\ [m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}}\end{pmatrix}$
Data Process
Operates division and encrypts computational result after decryption by calling $PDec2$
$\left[\left\lfloor\frac{m_1}{m_2}\right\rfloor + r_2\right]_{pk_{CP}}$
Data Reprocess
Removes the mask and generates a key pair (ck_1, pk_{ck_1}), and sets $c_1 = {ck_1}^{-1} mod\ n$
$\left[c_1\left\lfloor\frac{m_1}{m_2}\right\rfloor\right]_{pk_{CP}}$
Data Reprocess
Performs decryption with own secret key and encrypts result with another public key
$\left[c_1\left\lfloor\frac{m_1}{m_2}\right\rfloor\right]_{pk_{ck_2}}$
Additional Process
Performs exponential computation
$\left\{\left[c_1\left\lfloor\frac{m_1}{m_2}\right\rfloor\right]_{pk_{ck}}, CK\right\}$
Data Access
Decrypt CK to get ck for decryption

Fig. 3. The procedure of division computation with flexible access control.

Eq. (18) by calling HE^{ABE} algorithm.

$$\left[\left\lfloor\frac{m_1}{m_2}\right\rfloor\right]_{pk_{ck}} = \left(\left[c_1\left\lfloor\frac{m_1}{m_2}\right\rfloor\right]_{pk_{ck_2}}\right)^{1,ck_1} \tag{17}$$

$$CK = CK_1 * CK_2 \tag{18}$$

Finally, DSP keeps $[\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{ck}}$ and CK for user access.

Step 8 (Data Access @ DRs): Upon receiving the computational result and CK from DSP, the DRs who satisfy the access policy can obtain the secret key SK' from Authority. Thus, the authorized DRs can obtain ck by calling $Dec^{ABE}()$ to decrypt CK, and further get the final quotient by calling $Dec([\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{ck}}, ck)$.

3) Privacy-Preserving Remainder Computation for a Targeted Data Requester (Scheme 3): Herein, we omit the same first three steps as in Scheme 1 and only introduce the additional part as below.

Step 4 (Data Process @ CP): Once getting the data forwarded by DSP, CP first calls $PDec2(*, sk_{CP})$ to obtain concealed plaintext and seperately employs Eqs. (13) and (19) to get masked quotient and remainder through computation on plaintext.

$$Rr_1 = (m_1 r_1 + m_2 r_1 r_2) - m_2 r_1 * \left(\left\lfloor\frac{m_1}{m_2}\right\rfloor + r_2\right) \tag{19}$$

Then CP calls $Enc(*, pk_{DR})$ to encrypt the above computational result as $[\lfloor\frac{m_1}{m_2}\rfloor + r_2]_{pk_{DR}}$ and $[Rr_1]_{pk_{DR}}$, and sends the ciphertext to DSP.

Step 5 (Data Additional Process @ DSP): DSP removes the mask from received ciphertext to get encrypted quotient and remainder by applying Eqs. (14) and (20) respectively.

$$[R]_{pk_{DR}} = \left([Rr_1]_{pk_{DR}}\right)^{r_1^{-1}}. \tag{20}$$

Step 6 (Data Access @ DR): Upon receiving the computational results from DSP, the targeted DR can decrypt the ciphertext $[\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{DR}}$ and $[R]_{pk_{DR}}$ to get the final quotient and remainder of the division by calling $Dec(*, sk_{DR})$.

4) Privacy-Preserving Remainder Computation With Flexible Access Control (Scheme 4): We introduce its details below by omitting the same first three steps as in Scheme 2.

Step 4 (Data Process @ CP): CP decrypts the data packet from DSP by invoking $PDec2(*, sk_{CP})$ and obtains two messages $m_2 r_1$ and $(m_1 r_1 + m_2 r_1 r_2)$. Then it performs basic operations to get $\lfloor\frac{m_1}{m_2}\rfloor + r_2$ and Rr_1. Furthermore, CP calls $Enc(*, pk_{CP})$ to encrypt the computational results and sends the encrypted data packet $\{[\lfloor\frac{m_1}{m_2}\rfloor + r_2]_{pk_{CP}}, [Rr_1]_{pk_{CP}}\}$ to DSP.

Step 5 (Data Reprocess @ DSP): DSP first selects a partial secret key ck_1 and sets a random number c_1 as $(ck_1)^{-1}$ mod n. Then it removes the mask from received ciphertext to get encrypted quotient with Eq. (15) and encrypted remainder with Eq. (21). Moreover, DSP uses Eqs. (16) and (22) to conceal the quotient and remainder from CP respectively.

$$[R]_{pk_{CP}} = \left([Rr_1]_{pk_{CP}}\right)^{{r_1}^{-1}} \tag{21}$$

$$[c_1 R]_{pk_{CP}} = \left([R]_{pk_{CP}}\right)^{c_1} \tag{22}$$

Next, the data packet $\{[c_1\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{CP}}, [c_1R]_{pk_{CP}}\}$ is sent to CP.

Step 6 (Data Reprocess @ CP): With received data packet, CP first performs $PDec2(*, sk_{CP})$ on encrypted data. Then, it chooses a partial key ck_2 to generate a key pair (ck_2, pk_{ck_2}) and calls $Enc(*, pk_{ck_2})$ to encrypt the masked data. Detailed processes are described as below.

1) $[c_1\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{CP}} \xrightarrow{PDec2(*,sk_{CP})} c_1\lfloor\frac{m_1}{m_2}\rfloor \xrightarrow{Enc(*,pk_{ck_2})} [c_1\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{ck_2}}$;
2) $[c_1R]_{pk_{CP}} \xrightarrow{PDec2(*,sk_{CP})} c_1R \xrightarrow{Enc(*,pk_{ck_2})} [c_1R]_{pk_{ck_2}}$;

In addition, CP calls $Enc^{ABE}(ck_2, \mathcal{T}, PK')$ to encrypt ck_2 as CK_2.

The data packet $\{[c_1\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{ck_2}}, [c_1R]_{pk_{ck_2}}, CK_2\}$ is sent to DSP.

Step 7 (Additional Process @ DSP): Upon receiving data packet from CP, DSP performs exponential operations according to Eq. (11), and applys Eqs. (17) and (23) to obtain final encrypted quioent and remainder respectively. In addition, it calls $Enc^{ABE}(ck_1, \mathcal{T}, PK')$ to get CK_1 and obtains the access key *CK* through Eq. (18).

$$[R]_{pk_{ck}} = \left([c_1R]_{pk_{ck_2}}\right)^{1,ck_1} \tag{23}$$

Finally, DSP keeps the encrypted data packet $\{[\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{ck}}, [R]_{pk_{ck}}\}$ and the key *CK* for user access.

Step 8 (Data Access @ DR): The DRs whose attributes meet the access policy can get a secret key SK' from Authority, which can be used to get *ck* by calling $Dec^{ABE}(PK', SK', CK)$. Then DRs decrypt the received ciphertext $[\lfloor\frac{m_1}{m_2}\rfloor]_{pk_{ck}}$ and $[R]_{pk_{ck}}$ obtained from DSP to get the final quotient and remainder of the division computation.

C. Scheme Extension

In this section, we extend the above schemes to support various types of numbers including fixed-point numbers and fractional numbers.

1) Division Over Fixed Point Numbers: We extend the above division scheme to obtain a floating result with a fixed number of digits after the decimal point. Assuming that the length of the fractional field is *k*, DP should first scale the numerator m_1 to m_1' through Eq. (24). DSP receives the encrypted data $\{[m_1'], [m_2]\}$ and processes the data cooperating with CP. The DR requests the final result and decrypts it as $\lfloor\frac{m_1'}{m_2}\rfloor$, then the result with fixed number of digits can be computed by Eq. (25).

$$m_1' = m_1 * 2^k \tag{24}$$

$$Q = \left\lfloor\frac{m_1'}{m_2}\right\rfloor * 2^{-k} \tag{25}$$

Herein, we can evaluate deviation of the computation result from the actual division result with Eq. (26).

$$\delta = m_1 - Q * m_2. \tag{26}$$

2) Secure Computations on Fractional Numbers: Given two fractional numbers $m_{1,1}/m_{1,2}$ and $m_{2,1}/m_{2,2}$, we can perform following computations based on the above proposed division schemes, multiplication and addition proposed in our previous work [1].

a) Addition over fractional numbers: The addition of two fractional numbers can be represented as Eq. (27).

$$\frac{m_{1,1}}{m_{1,2}} + \frac{m_{2,1}}{m_{2,2}} = \frac{m_{1,1} * m_{2,2} + m_{2,1} * m_{1,2}}{m_{1,2} * m_{2,2}} \tag{27}$$

When DSP receives the encrypted data packet $\{[m_{1,1}], [m_{1,2}], [m_{2,1}], [m_{2,2}]\}$, it first interacts with CP to perform privacy-preserving multiplication to gain ciphertexts $[m_{1,1} * m_{2,2}]$, $[m_{2,1} * m_{1,2}]$ and $[m_{1,2} * m_{2,2}]$, then computes $[m_{1,1} * m_{2,2}. + m_{2,1} * m_{1,2}]$ through additive homomorphism. Finally, DSP and CP cooperate to perform corresponding division computation on the two ciphertexts $[m_{1,1} * m_{2,2} + m_{2,1} * m_{1,2}]$ and $[m_{1,2} * m_{2,2}]$ by applying Scheme 1-4 according to concrete scenarios.

b) Multiplication over fractional numbers: The product of two encrypted factional numbers can be computed as $[\frac{m_{1,1}*m_{2,1}}{m_{1,2}*m_{2,2}}]$. Thus, DSP and CP first perform secure multiplication to get $[m_{1,1}*m_{2,1}]$ and $[m_{1,2}*m_{2,2}]$. Then, an encrypted division result can be acquired with the cooperation of DSP and CP through division computation.

c) Division over two fractional numbers: The division calculation over the two fractional numbers is equivalent to $m_{1,1} * m_{2,2}/m_{1,2} * m_{2,1}$. After obtaining ciphertext products $[m_{1,1} * m_{2,2}]$ and $[m_{1,2} * m_{2,1}]$ by conducting multiplication over encrypted data, DSP executes the division computation over the two encrypted data by cooperating with CP.

V. Security Analysis and Performance Evaluation

A. Correctness Proof of Multiplicative Homomorphism of CP-ABE

Herein, we prove the multiplicative homomorphism of CP-ABE. The ciphertext of two messages M_1 and M_2 under the same ABE access policy $\mathcal{T}$ can be obtained with Eq. (28):

$$\begin{aligned} Enc(M_i) = \big\{&M_i e(g,g)^{\alpha s_i}, C_i = h^{s_i}, \forall y_i \in Y: C_{y_i} = g^{q_{y_i}}(0), \\ &C'_{y_i} = H(att(y_i))^{q_{y_i}(0)}\big\}, (i = 1, 2) \end{aligned} \tag{28}$$

Then we can get the product of two ciphertexts through Eq. (29).

$$\begin{aligned} &Enc(M_1) * \ Enc(M_2) \\ &= \Big\{M_1 * M_2 e(g,g)^{\alpha(s_1+s_2)}, C = h^{s_1+s_2}, \forall y \in Y: \\ &\quad C_y = g^{q_{y_1}(0)+q_{y_2}(0)}, C'_y = H(\ att(y))^{q_{y_1}(0)+q_{y_2}(0)}\Big\} \end{aligned} \tag{29}$$

1) For the leaf node *y* from the access tree $\mathcal{T}$, if the attribute $i \in \mathcal{S}$, the recursive decryption algorithm can be defined as follows.

$$\begin{aligned} &DecryptNode\big(Enc(M_1) * Enc(M_2), SK', y\big) \\ &= \frac{e(D_i, C_y)}{e(D'_i, C'_y)} = \frac{e\Big(g^r \cdot H(i)^{r_i}, h^{q_{y_1}(0)+q_{y_2}(0)}\Big)}{e\Big(g^{r_i}, H(i)^{q_{y_1}(0)+q_{y_2}(0)}\Big)} \\ &= e(g,g)^{r\left(q_{y_1}(0)+q_{y_2}(0)\right)} \end{aligned} \tag{30}$$

2) For the non-leaf node x from the access tree $\mathcal{T}$, $DecryptNode(Enc(M_1) * Enc(M_2), SK', x)$ calls $DecryptNode(Enc(M_1) * Enc(M_2), SK', y)$ for all nodes y that are children of x to perform decryption. Then, we can get Eq. (31) according to the linearity of the access structure.

$$\begin{aligned} &DecryptNode\big(Enc(M_1) * Enc(M_2), SK', x\big) \\ &= e(g, g)^{r\left(q_{x_1}(0)+q_{x_2}(0)\right)} \end{aligned} \tag{31}$$

Thus, for the root node of the access tree $\mathcal{T}$, we can get Eq. (32) as follows.

$$\begin{aligned} &DecryptNode\big(Enc(M_1) * Enc(M_2), SK', r\big) \\ &= e(g, g)^{rq_R(0)} = e(g, g)^{r(s_1+s_2)}. \end{aligned} \tag{32}$$

Finally, the ciphertext can be decrypted as Eq. (33).

$$\begin{aligned} &\frac{M_1 * M_2 e(g, g)^{\alpha(s_1+s_2)}}{\frac{e\left(h^{s_1+s_2}, g^{(\alpha+r)/\beta}\right)}{e(g,g)^{r(s_1+s_2)}}} \\ &= \frac{M_1 * M_2 e(g, g)^{\alpha(s_1+s_2)} \cdot e(g, g)^{r(s_1+s_2)}}{e(g, g)^{(\alpha+r)(s_1+s_2)}} \\ &= M_1 * M_2 \end{aligned} \tag{33}$$

In summary, we can get the conclusion shown in Eq. (34).

$$Dec(Enc(M_1) * \ Enc(M_2)) = M_1 * M_2. \tag{34}$$

B. Correctness Proof of Division Schemes

Here, we take Scheme 1 as an example to prove its correctness. As its encryption and decryption restricts the length of data, we should guarantee that Scheme 1 can get the correct quotient from ciphertext.

Upon data received from DSP, CP can decrypt them to get $m_2 r_1 \bmod n$ and $m_1 r_1 + m_2 r_1 r_2 \bmod n$ where $m_2 r_1 < n$ and $m_1 r_1 + m_2 r_1 r_2 < n$ owing the limitation to the length of data selected. Hence, we can get Eqs. (36) and (37) based on the assumption as Eq. (35).

$$m_1 = A * m_2 + R, \mathit{where}\ R < m_2 \tag{35}$$

$$(m_1 r_1 + m_2 r_1 r_2) = A m_2 r_1 + R r_1 + m_2 r_1 r_2 \tag{36}$$

$$(m_1 r_1 + m_2 r_1 r_2)/(m_2 r_1) = A + R/m_2 + r_2 \tag{37}$$

As $R < m_2$, then R/m_2 is smaller than 1. Therefore, we can get Eq. (38), which is the quotient.

$$\left\lfloor \frac{m_1}{m_2} \right\rfloor = A. \tag{38}$$

C. Security Analysis

Similar to our previous work [1], the security of all schemes in this paper inherits from the semantic security of HRES in [31] and ABE. To prove their security, we follow the security model and attack model with the existence of four semi-honest adversaries. Except the Authority, all other entities may be compromised. Hence, we construct four simulators $(Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$ to fight against their corresponding adversaries $(\mathcal{A}_{DP}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DR})$ that compromise *DP*, *DSP*, *CP* and *DR*, respectively.

Scheme 1 can securely obtain the quotient from encrypted data through the cooperation between two servers of DSP and CP in the existence of semi-honest adversaries $(\mathcal{A}_{DP}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DR})$.

Here we construct four simulators including Sim_{DP}, Sim_{DSP}, Sim_{CP} and Sim_{DR}.

Sim_{DP} simulates $\mathcal{A}_{DP}$: Sim_{DP} only needs to outsource its data by calling $EncTK(m_i, PK)$, hence its security can directly inherit from the original HRES. Though DP may colludes with one server (for example, DSP), $\mathcal{A}_{DSP}$ can only get the partial decryption result $[m_i]_{pk_{CP}}$ through $PDec1([m_i], sk_{DSP})$ in Step 3. Finally, $\mathcal{A}_{DP}$ can get the ciphertext $[m_i]_{pk_{CP}}$ and $[m_i]$. Owing to the security of HRES, $\mathcal{A}_{DP}$ cannot get anything from the data outsourced from other users.

Sim_{DSP} simulates $\mathcal{A}_{DSP}$ as follows: first Sim_{DSP} calls $EncTK(*, PK)$ to encrypt random messages $\widetilde{m_1}$ and $\widetilde{m_2}$; then it chooses some random numbers to obtain $[\widetilde{m_2} r_1]$, $[\widetilde{m_1} r_1 + \widetilde{m_2} r_1 r_2]$ and then further decrypts them into $[\widetilde{m_2} r_1]_{pk_{CP}}, [\widetilde{m_1} r_1 + \widetilde{m_2} r_1 r_2]_{pk_{CP}}$ by calling $PDec1(*, sk_{DSP})$. In Step 5, it receives $[\lfloor\frac{\widetilde{m_1}}{\widetilde{m_2}}\rfloor + r_2]_{pk_{DR}}$ by accessing Sim_{CP}, which is encrypted with the public key of the targeted DR. Then it use the additive homomorphism to remove the mask and obtain $[\lfloor\frac{\widetilde{m_1}}{\widetilde{m_2}}\rfloor]_{pk_{DR}}$. Finally, Sim_{DSP} outputs $\{[\widetilde{m_2} r_1], [\widetilde{m_1} r_1 + \widetilde{m_2} r_1 r_2].,[\widetilde{m_2} r_1]_{pk_{CP}}, [\widetilde{m_1} r_1 + \widetilde{m_2} r_1 r_2]_{pk_{CP}}, [\lfloor\frac{\widetilde{m_1}}{\widetilde{m_2}}\rfloor + r_2]_{pk_{DR}}, [\lfloor\frac{\widetilde{m_1}}{\widetilde{m_2}}\rfloor]_{pk_{DR}}\}$ to Sim_{DSP}. If Sim_{DSP} replies with $\perp$, Sim_{DSP} returns $\perp$.

The views of Sim_{DSP} are merely the ciphertexts under the public keys of DR and CP. Though the Sim_{DSP} may collude with the DR, it can only get the raw data from the compromised DR rather than the challenged users. Hence, Sim_{DSP} cannot get any information about the division and original data owing to intrinsic security of HRES and the honesty of challenged cloud users. As DSP chooses random numbers in operations, Sim_{DSP} still cannot obtain any other information by analyzing the results obtained from several challenges.

Sim_{CP} simulates $\mathcal{A}_{CP}$ as follows: Sim_{CP} accesses Sim_{DSP} to get the ciphertexts $[\widetilde{m_2} r_1]_{pk_{CP}}$, $[\widetilde{m_1} r_1 + \widetilde{m_2} r_1 r_2]_{pk_{CP}}$. Then it decrypts them to get the masked raw data $\lfloor\frac{\widetilde{m_1}}{\widetilde{m_2}}\rfloor + r_2$ and $[\lfloor\frac{\widetilde{m_1}}{\widetilde{m_2}}\rfloor + r_2]_{pk_{DR}}$. Finally, it sends them to $\mathcal{A}_{CP}$. If $\mathcal{A}_{CP}$ replies with $\perp$, Sim_{CP} returns $\perp$. Owing to the security of HRES and the random numbers, the security can be guaranteed that $\mathcal{A}_{CP}$ can get nothing.

Sim_{DR} simulates $\mathcal{A}_{DR}$ as follows: Besides the challenged data, any random ciphertexts are chosen and decrypted to gain the original data. And Sim_{DR} sends them to $\mathcal{A}_{DR}$. But owing to the semantic security of HRES and the random numbers selected for each encryption, it guarantees the indistinguishability of the ciphertext of challenged data and the random message.

Similar security proof to Scheme 1 can be performed for Scheme 2-4. The proofs of Scheme 2 and Scheme 4 are a bit different from Scheme 1, but their security can be directly guaranteed by HRES and CP-ABE.

TABLE II
COMPUTATIONAL COMPLEXITY ANALYSIS

Entity	Scheme	Computations	Complexity
DP	All Schemes	$2{*}ModExp$	$\mathcal{O}(1)$
DSP	Scheme 1	$5*ModExp+1*ModMul$	$\mathcal{O}(1)$
	Scheme 2	$7*ModExp+1*ModMul+(2\|S\|+1)Exp+\|S\|Hash+\|S\|Mul$	$\mathcal{O}(\|S\|)$
	Scheme 3	$6*ModExp+1*ModMul$	$\mathcal{O}(1)$
	Scheme 4	$10*ModExp+1*ModMul+(2\|S\|+1)Exp+\|S\|Hash+\|S\|Mul$	$\mathcal{O}(\|S\|)$
CP	Scheme 1	$4*ModExp$	$\mathcal{O}(1)$
	Scheme 2	$7*ModExp+1*ModMul+(2\|S\|+1)Exp+\|S\|Hash$	$\mathcal{O}(\|S\|)$
	Scheme 3	$6*ModExp$	$\mathcal{O}(1)$
	Scheme 4	$10*ModExp+1*ModMul+(2\|S\|+1)Exp+\|S\|Hash$	$\mathcal{O}(\|S\|)$
DR	Scheme 1	$1*ModExp$	$\mathcal{O}(1)$
	Scheme 2	$1*ModExp+\vartheta*BiPair$	$\mathcal{O}(\vartheta)$
	Scheme 3	$2*ModExp$	$\mathcal{O}(1)$
	Scheme 4	$2*ModExp+\vartheta*BiPair$	$\mathcal{O}(\vartheta)$
Authority	Scheme 2 and Scheme 4	$(2\|S\|+4)Exp+\|S\|Hash$	$\mathcal{O}(\|S\|)$

Notes:$ModExp$: modular exponentiation; $ModMul$: modular multiplication; Exp: exponentiation in ABE; Mul: multiplication in ABE; $|S|$: the numbers of attributes in access policy $\mathcal{T}$; ϑ: the number of attributes needed to satisfy $\mathcal{T}$.

D. Performance Evaluation

To show the performance of our schemes, we first give the analysis of computational complexity of CP-ABE and HRES algorithms, followed by the computational complexity of our proposed division schemes. Furthermore, we tested their efficiency and scalability by extensive simulations.

1) Computational Complexity: We assume that there are |*S*| attributes embedded in access tree $\mathcal{T}$ and that access policy needs at most ϑ attributes to be satisfied for successful decryption. Due to paper length limitation, we omit analysis details. The computational complexity is shown in Table II.

2) Experimental Results: We further simulated the proposed four division schemes and tested their performance to verify aforementioned theoretical analysis. The simulations were performed in a desktop computer with Intel Core i3-3240 CPU 3.4 GHz and 4GB RAM with jPBC library. To ensure higher accuracy, we performed each test at least 200 times and recorded the average values of consumed time. Unless specifically stated, we set the length of ck_1 and ck_2 as 255 bits, the length of m_1 as 255 bits, the length of m_2 as 250 bits and the length of random number $L(r_i)$ as 255 bits. In our test machine, one bilinear pairing costs about 7 milliseconds.

By changing the length of *n*, we first analyzed the efficiency of data processing in each step of four division schemes. Then we verified the scalability of our proposed scheme with different length of input data $L(m_1)$ and $L(m_2)$, where we keep difference between $L(m_1)$ and $L(m_2)$ as 5 bits ($L(m_1) > L(m_2)$). Finally, we compared with existing works to show advantages of our schemes.

a) Efficiency of data processing:

Test 1 (Efficiency of CP-ABE for access control): First of all, we tested the efficiency of CP-ABE with different numbers of attributes involved in policy, which vary from 2 to 10 as shown in Fig. 4. $Setup^{ABE}$ does not vary with the number of involved attributes, while the computation cost of $Enc^{ABE}()$ and $keyGen^{ABE}()$ grow with increased number of attributes. One attribute should be satified in the policy tree, which indicates the operation time of $Dec^{ABE}()$ is constant. In addition, $HE^{ABE}()$ only takes less than 1 ms, which is not showed in Fig. 4.

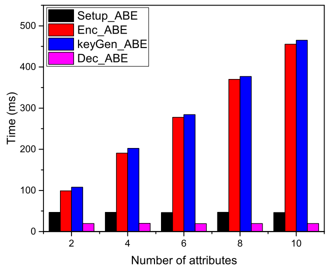

Fig. 4. Cost of CP-ABE with different number of attributes.

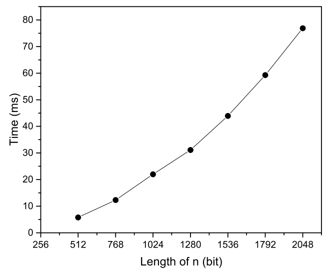

Fig. 5. Operation time of DPs with different length of *n*.

Test 2 (Influence of the length of n on performance): Fig. 5 presents the execution time of DP when the length of *n* is set to different values, which is similar in all division schemes.

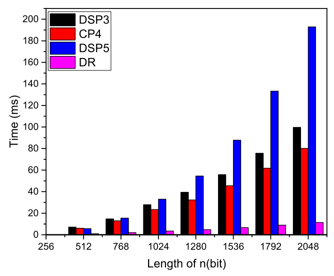

Fig. 6. Execution time of scheme 1 with different n.

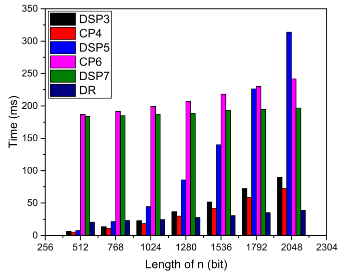

Fig. 7. Execution time of scheme 2 with different n.

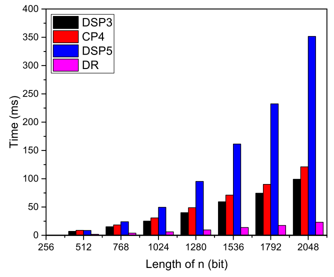

Fig. 8. Execution time of scheme 3 with different n.

It is observed that the data processing on DP is efficient and applicable in devices with limited resources.

Scheme 1 and Scheme 3 performs four steps except for the first two steps of system setup and data uploading. Their operation time are shown in Fig. 6 and Fig. 8 respectively, which indicate the increasing computation time of DSP in Step 3 (DSP3), CP in Step 4 (CP4), DSP in Step 5 (DSP5) and DR in Step 6 (DR6) with the growth of the length of n. We can observe that data processing and computation of DSP in both schemes are similar in Step 3. However, the data processing in other steps of Scheme 3 that contain the calculations for getting the remainder is more time-consuming than that in Scheme 1, which conforms with our aforementioned complexity analysis in Section V-D1. The data processing of DSP in Step 5 is the most time-consuming, which costs about 190 milliseconds (ms) in Scheme 1 and 350ms in Scheme 3 when the length of n is 2048 bits. While operating other steps takes about 100ms.

Scheme 2 and Scheme 4 introduce CP-ABE to support flexible access control and add a round of interaction between DSP and CP, their operation time of each step is shown in Fig. 7 and Fig. 9 respectively. In this test, we set the number of attributes as 4. We can observe that data processing time of DSP in Step 3 is also similar to Scheme 1 and Scheme 3. In

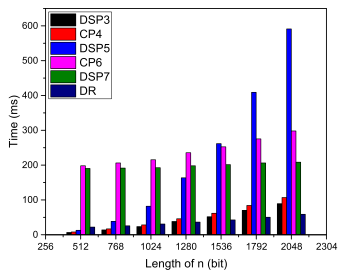

Fig. 9. Execution time of scheme 4 with different n.

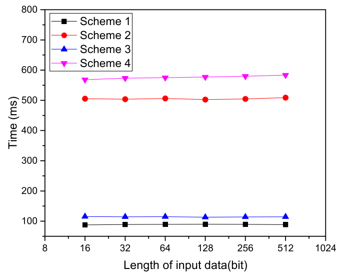

Fig. 10. Execution time of four schemes with different length of provided data.

TABLE III
THE COMPARISON OF OUR PROPOSED SCHEMES

Scheme	Computational Results	Computation Cost (ms)	Flexible Access Control
Scheme 1	Quotient	88	N
Scheme 2	Quotient	505	Y
Scheme 3	Quotient and Remainder	114	N
Scheme 4	Quotient and Remainder	577	Y

Notes: Y: Supported; N: Unsupported.

addition, the computational cost of added two steps are about 200ms in both Scheme 2 and Scheme 4. the computation time of DSP in Step 5 is still the most time-consuming, which costs about 300ms in Scheme 2 and 600ms in the Scheme 4 when the length of n is 2048 bits.

In all four schemes, the computational cost of DR is less than 50ms, which is well accepted by cloud users with constrained resources.

In summary, most computational costs are undertaken by two servers while the computational overheads of cloud users are acceptable, which implies the efficiency and practicality of our proposed schemes.

b) Scalability of proposed schemes:

Test 3 (Influence of length of provided data on data processing): In this experiment, we tested the whole execution time from data uploading to data access of each scheme with different lengths of provided raw data, which varies from 16 to 512 bits. From Fig. 10, we can observe that the operation time do not vary with the changing bit length of data, which indicates that our proposed division schemes are applicable for both normal value data and big value data.

TABLE IV
THE COMPARISON OF OUR WORK WITH EXISTING WORKS

Ref	Applied Technologies or Methods	Computational Results	Data Type	Over head	PP	FAC
[16]	Arithmetic transformations	Quotient	Non-integers	high	Y	N
[17]	Taylor series	Quotient	Integers	high	Y	N
[18]	Arithmetic transformations	Quotient	Integers	low	N	N
[23], [24]	Bit decomposition	Quotient and Remainder	Integers	high	Y	N
[19]	Goldschmidt's method	Quotient	Float point numbers	high	N	N
[21]	Iteration	Quotient	Float point numbers	high	Y	N
Our Work	CP-ABE and HRES	Quotient and Remainder	Integers, fractional numbers and fixed-point numbers	low	Y	Y

Notes: PP: Privacy Preservation; FAC: Flexible Access Control; Y: Supported; N: Unsupported.

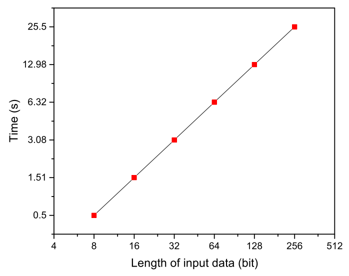

Fig. 11. Execution time of SBD in [22] with different length of provided data.

Based on the experimental result shown in Fig. 10 and the analysis in Table II, we can find that Scheme 1 and Scheme 3 are more efficient than other two schemes and that Scheme 1 is a little more efficient than Scheme 3. But Scheme 1 only provides the quotient. Compared with Scheme 1 and Scheme 3, Scheme 2 and Scheme 4 incur higher computational costs, but they enable flexible access control by adopting ABE. Similarly, Scheme 4 incurs a little higher computational cost than Scheme 2, but it provides the remainder of the division. A brief comparison of the four schemes is given in Table III.

c) Comparison with existing work:

Test 4 (Performance comparison with an existing secure division protocol in [24]): Before comparing with existing work in [24], we tested the execution time of SBD protocol in [22] with different lengths of original provided data, which varies from 8 to 256 bits as shown in Fig. 11. We can observe that the computation cost of SBD grows fast with the increasing bit length of data and that it needs up to 25s to decompose 256-bit data. Thus, our scheme is superior to the division protocol based on SBD [22] in terms of processing large integers. Herein, we set the length of input data l as 10 bits and compared the cost of DSP and CP in Scheme 4 with the cost of CSP and CP to compute encrypted quotient and remainder in existing work [24], which vary with the bit length of n. From Fig.12, we can find that Scheme 4 as the most time-consuming scheme in the four proposed schemes is much more efficient than the existing division scheme over encrypted integer.

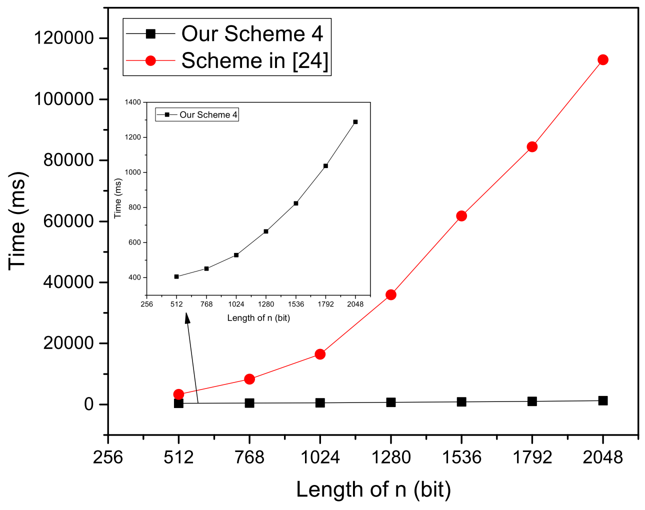

Fig. 12. Operation time of our scheme 4 compared with existing work.

Based on aforementioned discussion and evaluation, we summarize the comparison results of our work with existing work in Table IV to demonstrate its superiority.

VI. CONCLUSION

In this paper, we proposed four privacy-preserving division schemes over integers with flexible access control and extended them to support computations over encrypted fractional numbers and fixed-point numbers. We seriously analyzed the correctness and security of our schemes. Through experimental simulations and comparison with existing work, we further showed the efficiency and scalability of our schemes. Thus, we greatly extended the framework of privacy-preserving computation with flexible access control [1] by offering one missed important computation–division and additionally supporting computations over encrypted fractional numbers and fixed-point numbers. In the future, we will apply our schemes into real application scenarios to demonstrate their practicality.

References

[1] W. Ding, Z. Yan, and R. H. Deng, "Privacy-preserving data processing with flexible access control," *IEEE Trans. Depend. Secure Comput.*, to be published.

[2] D. Liu, Z. Yan, W. Ding, and M. Atiquzzaman, "A survey on secure data analytics in edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4946–4967, Jun. 2019.

[3] W. Ding, X. Jing, Z. Yan, and L. T. Yang, "A survey on data fusion in Internet of Things: Towards secure and privacy-preserving fusion," *Inf. Fusion*, vol. 51, pp. 129–144, Nov. 2019.

[4] R. Hu, Z. Yan, W. X. Ding, and L. T. Yang, "A Survey on Data Provenance in IoT," *World Wide Web J.*, 2019.

[5] X. Yu, Z. Yan, and R. Zhang, "Verifiable outsourced computation over encrypted data," *Inf. Sci.*, vol. 479, pp. 372–385, Apr. 2019.

[6] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.

[7] B. Wang, M. Li, S. S. M. Chow, and H. Li, "A tale of two clouds: Computing on data encrypted under multiple keys," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, San Francisco, CA, USA, 2014, pp. 337–345.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput. (STOC)*, Bethesda, MD, USA, May/Jun. 2009, pp. 169–178.

[9] M. Yagisawa, "Fully homomorphic encryption without bootstrapping," *IACR Cryptol. ePrint Archive*, vol. 2015, p. 474, Mar. 2015.

[10] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surveys*, vol. 51, no. 4, pp. 1–35, 2018.

[11] V. C. Hu, T. Grance, D. F. Ferraiolo, and D. R. Kuhn, "An access control scheme for big data processing," in *Proc. 10th IEEE Int. Conf. Collabarative Comput. Netw. Appl. Worksharing (CollaborateCom)*, Miami, FL, USA, Oct. 2014, pp. 1–7.

[12] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. 9th Int. Symp. Privacy Enhanc. Technol. (PETS)*, Seattle, WA, USA, Aug. 2009, pp. 235–253.

[13] E. Kiltz, G. Leander, and J. Malone-Lee, "Secure computation of the mean and related statistics," in *Proc. 2nd Theory Cryptography Conf. Theory Cryptography (TCC)*, Cambridge, MA, USA, Feb. 2005, pp. 283–302.

[14] P. Bunn and R. Ostrovsky, "Secure two-party *k*-means clustering," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Alexandria, VA, USA, Oct. 2007, pp. 486–497.

[15] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, "Privacy enhanced recommender system," in *Proc. 31st Symp. Inf. Theory Benelux*, 2010, pp. 35–42.

[16] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schröder, "Secure computations on non-integer values," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, Seattle, WA, USA, Dec. 2010, pp. 1–6.

[17] M. Dahl, C. Ning, and T. Toft, "On secure two-party integer division," in *Proc. 16th Int. Conf. Financial Cryptography Data Security (FC)*, Mar. 2012, pp. 164–178.

[18] T. Veugen, "Encrypted integer division and secure comparison," *Int. J. Appl. Cryptography*, vol. 3, no. 2, pp. 166–180, 2014.

[19] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *Proc. 14th Int. Conf. Financ. Cryptography Data Security (FC)*, Jan. 2010, pp. 35–50.

[20] R. Bhoyar, P. Palsodkar, and S. Kakde, "Design and implementation of Goldschmidts algorithm for floating point division and square root," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2015, pp. 1588–1592.

[21] C. Ugwuoke, Z. Erkin, and R. L. Lagendijk, "Secure fixed-point division for homomorphically encrypted operands," in *Proc. 13th Int. Conf. Availability Rel. Security (ARES)*, Hamburg, Germany, Aug. 2018, pp. 1–10.

[22] B. K. Samanthula, C. Hu, and W. Jiang, "An efficient and probabilistic secure bit-decomposition," in *Proc. 8th ACM Symp. Inf. Comput. Commun. Security (AsiaCCS)*, Hangzhou, China, May 2013, pp. 541–546.

[23] J. Feng, L. T. Yang, Q. Zhu, and K.-K. R. Choo, "Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment," *IEEE Trans. Depend. Secure Comput.*, to be published.

[24] X. Liu, K.-K. R. Choo, R. H. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 1, pp. 27–39, Jan./Feb. 2018.

[25] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Security (CCS)*, Alexandria, VA, USA, Nov. 2006, pp. 89–98.

[26] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[27] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[28] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. 29th IEEE Int. Conf. Comput. Commun. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, San Diego, CA, USA, Mar. 2010, pp. 534–542.

[29] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy (SP)*, Oakland, CA, USA, May 2007, pp. 321–334.

[30] H. Cui, R. H. Deng, Y. Li, and B. Qin, "Server-aided revocable attribute-based encryption," in *Proc. Eur. Symp. Res. Comput. Security*, 2016, pp. 570–587.

[31] W. Ding, Z. Yan, and R. H. Deng, "Encrypted data processing with homomorphic re-encryption," *Inf. Sci.*, vol. 409, pp. 35–55, Oct. 2017.

Wenxiu Ding received the B.Eng. and Ph.D. degrees in information security from Xidian University, Xi'an, China, in 2012 and 2017, respectively. From 2015 to 2016, she was a Research Assistant with the School of Information Systems, Singapore Management University. She is currently a Lecturer with the School of Cyber Engineering, Xidian University. Her research interests include RFID authentication, privacy preservation, data mining, and trust management.

Rui Hu received the B.Eng. degree in information security from Xidian University, Xi'an, China, in 2017, where she is currently pursuing the master's degree with the State Key Laboratory on Integrated Services Networks. Her research interests are in data provenance, big data security, and privacy preservation.

Zheng Yan received the B.Eng. degree in electrical engineering and the M.Eng. degree in computer science and engineering from Xi'an Jiaotong University, Xi'an, China, in 1994 and 1997, respectively, the second M.Eng. degree in information security from the National University of Singapore, Singapore, in 2000, and the Licentiate of Science and the Doctor of Science in Technology degrees in electrical engineering from the Helsinki University of Technology, Helsinki, Finland. She is currently a Professor with the Xidian University, Xi'an, China, and a Visiting Professor with Aalto University, Espoo, Finland. Her research interests are in trust, security, privacy, and security-related data analytics. She serves as a general or program chair for over 30 international conferences and workshops. She is a steering committee Co-Chair of the IEEE Blockchain international conference. She is also an Associate Editor of many reputable journals, including the IEEE INTERNET OF THINGS JOURNAL, *Information Sciences*, *Information Fusion*, the *Journal of Network and Computer Applications*, IEEE ACCESS, and *Security and Communication Networks*.

Xinren Qian received the B.Eng. degree in information security from Xidian University, Xi'an, China, in 2018, where he is currently pursuing the master's degree with State Key Laboratory on Integrated Servicces Networks. His research interests are in cloud storage and computing, and privacy preservation.

Robert H. Deng (F'16) is a AXA Chair Professor of cybersecurity and the Director of the Secure Mobile Centre, School of Information Systems, Singapore Management University (SMU). His research interests are in the areas of data security and privacy, network security, and system security. He received the Outstanding University Researcher Award from National University of Singapore, the Lee Kuan Yew Fellowship for Research Excellence from SMU, and the Asia–Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. He serves/served on many editorial boards and conference committees, including the editorial boards of *IEEE Security & Privacy Magazine*, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and the *Journal of Computer Science and Technology*, and the Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security. He is a fellow of Academy of Engineering Singapore.

Laurence T. Yang received the B.E. degree in computer science and technology from Tsinghua University, China, and the Ph.D. degree in computer science from the University of Victoria, Canada. He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, China, and also with the Department of Computer Science, St. Francis Xavier University, Canada. His research has been supported by the National Sciences and Engineering Research Council and the Canada Foundation for Innovation. His research interests include parallel and distributed computing and embedded and ubiquitous pervasive computing.

Mianxiong Dong received the B.S., M.S., and Ph.D. degrees in computer science and engineering from the University of Aizu, Aizuwakamatsu, Japan.

He is currently a Professor with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Japan. He was a JSPS Research Fellow with the School of Computer Science and Engineering, University of Aizu, and a Visiting Scholar with the BBCR Group, University of Waterloo, Waterloo, ON, Canada, supported by JSPS Excellent Young Researcher Overseas Visit Program, from April 2010 to August 2011. His research interests include wireless networks, cloud computing, and cyber-physical systems. He was selected as a Foreigner Research Fellow (a total of three recipients all over Japan) by NEC C&C Foundation in 2011. He was a recipient of the Best Paper Awards from the IEEE HPCC 2008, the IEEE ICESS 2008, the ICA3PP 2014, the GPC 2015, the IEEE DASC 2015, the IEEE VTC 2016-Fall, the FCST 2017, the 2017 IET Communications Premium Award, the IEEE TCSC Early Career Award in 2016, the IEEE SCSTC Outstanding Young Researcher Award in 2017, the 12th IEEE ComSoc Asia–Pacific Young Researcher Award in 2017, and the Funai Research Award in 2018. He has been serving as the Vice Chair for the IEEE Communications Society Asia/Pacific Region Meetings and Conference Committee, Leading Symposium Chair of the IEEE ICC in 2019, the Student Travel Grants Chair of IEEE GLOBECOM in 2019, and the Symposium Chair of the IEEE GLOBECOM in 2016 and 2017. He is an Editor for IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE NETWORK, IEEE WIRELESS COMMUNICATIONS LETTERS, IEEE CLOUD COMPUTING, IEEE ACCESS, as well as a Leading Guest Editor for *ACM Transactions on Multimedia Computing*, *Communications and Applications*, the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, and the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS.