Flexible Access Control over Privacy-Preserving Cloud Data Processing

Wenxiu Ding, Xinren Qian, Rui Hu, Zheng Yan, and Robert H. Deng

1 Introduction

Cloud computing has been widely adopted in various application domains owing to its specific advantages, which enables cloud users to store their data and perform various computations on the data without incurring a high cost. It even becomes the "lifeline" of many institutes or organizations. With the advent of Internet of Things, enormous amounts of data are produced and outsourced to the cloud for storage and analysis. Data analysis helps to gain insights on related entities in a physical world, which can provide tremendous value for various applications in multifarious domains, e.g., medical [1], cybersecurity education [2], and business [3]. However, the cloud may not be fully trusted by cloud users since it may reveal or disclose the data outsourced by the cloud users or the processed results of these data, which may seriously undermine user privacy. For example, to train the future generation or employees with cybersecurity skills, the customized cybersecurity exercises will be more suitable if more related information are collected and analyzed. But they may be reluctant to offer too much data (such as work culture, associated threats) due to privacy concern. Therefore, it has great significance to protect sensitive data and data processing results from being leaked to any unauthorized parties. A

W. Ding · X. Qian · R. Hu
School of Cyber Engineering, Xidian University, Xi'an, China

Z. Yan (✉)
School of Cyber Engineering, Xidian University, Xi'an, China

Department of Communications and Networking, Aalto University, Espoo, Finland
e-mail: zheng.yan@aalto.fi

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

K. Daimi, G. Francia III (eds.), *Innovations in Cybersecurity Education*,
https://doi.org/10.1007/978-3-030-50244-7_12

standard solution is to encrypt the data before uploading. However, data encryption introduces several challenges as described below.

First, encryption seriously restricts the computations/analysis over the outsourced data in the cloud. With traditional encryption algorithms (such as AES), it is impossible for the cloud to process the encrypted data directly. Some existing efforts adopted partially homomorphic encryption (PHE) to solve the problem, but they are limited only to multiplication and addition operations on encrypted data [4, 5], which are not sufficient to satisfy user demands in many applications. More operations, such as comparison and equality test, are required in practical applications [6, 7]. This requests further study on privacy-preserving computations. The basic operations can be widely applied to realize complex and useful applications, e.g., privacy-preserving classifications in machine learning [8], trust evaluation in Internet of Things [9], and medical analysis in e-health [10]. Obviously, the more basic computations available over encrypted data, the more support on complete and complex functions and algorithms. To realize arbitrary computations over ciphertext, schemes based on fully homomorphic encryption (FHE) were designed [11–13]; however, most FHE-based schemes suffer from huge computation overhead and high storage cost, which make them impractical for real-world deployment and wide usage.

Second, secure multi-user access control over processed results also needs to be supported [14]. Both existing PHE and FHE are single-user systems, which inherently lack support for multi-user access to the processing results of encrypted data. The scheme based on PHE in [15] supports distribution of addition operation results through an interactive protocol between two servers, but the protocol must be executed for each data request, thus is inefficient. Attribute-based encryption (ABE) is an effective tool to support fine-grained access control and multi-user access and has been applied in many application scenarios [16–18]. However, to our knowledge, there is no effort in the literature on fine-grained access control over the results of encrypted data computation. Previous work [19] aims to solve this problem by combining homomorphic encryption and proxy re-encryption, but it only supports one requester access at one time. In case multiple users want to access the same result, it needs to execute the designed scheme for each requester, which obviously incurs high communication and computation costs.

In this chapter, we propose a novel system in order to overcome the challenges as described above. It supports multiple basic computations over encrypted data and realizes flexible access control over the processing results by employing PHE and ABE, which can be easily extended and implemented to cybersecurity education. We present a family of protocols to efficiently realize several basic computations over encrypted data. Then, we extend the system with maximum, minimum, and division computations over integers. We propose to combine the ciphertext of ABE with homomorphism to realize a fine-grained access control of the processing results.

2 Related Work

With the development of cloud computing, cybersecurity education becomes critical because the traditional cybersecurity cannot guarantee the security of organizations due to the sophisticated networks, while it also becomes flexible by taking advantage of educational testbeds and framework. The cloud users (such as students and engineers) can be greatly benefited through cybersecurity education and get trained with enough technical skills. However, the risk of revealing personal data makes it urgent to enhance data security and user privacy.

Secure Data Processing Based on SMC

Secure multi-party computation (SMC) enables computations over multi-user outsourced data without revealing each input. It lays a technical foundation for many problems, such as database query, intrusion detection, and data mining with privacy preservation [9]. Several Schemes [20, 21] based on the popular SMC construction Sharemind [22] were proposed to achieve various secure computations. But the product of N pieces of data needs about 3^N multiplications of 32-bit numbers under the cooperation of three involved servers in Sharemind, which obviously cannot adapt to big data processing.

Secure Data Processing Based on Homomorphic Encryption

FHE Schemes [11–13] are designed to realize arbitrary computations over encrypted data. Due to their high computation overhead, some extended Schemes [23, 24] are proposed to improve efficiency. However, their computation and storage costs are still not satisfactory for practical applications [25, 26]. PHE can only support limited computations, but it is more efficient and practical than FHE and has been widely used in various applications. Some Schemes [4, 5] can only support addition and multiplication over a limited number of data inputs. In [4], decryption requires solving the problem of discrete logarithm, which seriously restricts the length and the number of data inputs. The multi-party computation framework proposed in [5] achieves addition and multiplication by following the idea of secret sharing. Similar to the SMC-based scheme in [21], it is unable to support the multiplication of a large number of data inputs. Liu et al. [6] proposed a framework for efficient outsourced data calculations with privacy preservation.

Secure Data Access Control

Cloud storage enables cloud users to upload their data to the cloud for storage and further sharing. However, it leads to a new problem that the cloud users lose full control over their data. Proxy re-encryption can also be adopted to manage data sharing in cloud [27, 28]. But it cannot support fine-grained access control on homomorphic computation result. Role-based access control (RBAC) can provide partial flexibility based on one level policy, which ensures that only the user with specified role can access the data. But, these constructions [29, 30] based on RBAC cannot support flexible access policies with various attribute structures.

ABE [31, 32] has been widely applied in cloud storage management for achieving fine-grained access control [33–35]. Furthermore, trust-based Schemes [16–18] simplify the attributes involved in ABE and take into consideration only trust levels. These schemes highly reduce the computation cost. But, only one entity is in charge of the access control, which makes this entity obviously knows the results.

Secure Division Based on Arithmetic Transformations

Katzenbeisser et al. [36] chose a tuple $(\rho_x, \sigma_x, \tau_x)$ to represent a value $x \in D_l$, which belongs to a certain interval $D_l = [-l; +l]$ with $l > 0$, where $\rho_x = 1$, σ_x encodes the sign of the value x and τ_x indicates the absolute of the value. The division result can be computed by basic operations on corresponding element through function $\text{LDIV}\left([\overline{x}], [\overline{y}]\right) = \left([\rho_x], [\sigma_x][\sigma_x], [\tau_x]\left[\tau_y\right]^{-1}\left[\tau_{C^2}\right]\right)$. Though the representation of numbers can support secure computations on non-integers, its 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. [37] 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. The implementation of several sub-protocols brings high computational overhead. Also, the frequent interactions bring high communication overhead.

Veugen [38] presented three protocols based on a client-server model where the client has encrypted data $[x]$ and the server has the corresponding decryption key K. In order to improve the precision of data analysis, Catrina and Saxena [39] attempted to approximately get a division result over two floating point numbers by applying the Goldschmidt method [40]. But this scheme cannot support division computations over encrypted input data. To overcome this issue, Ugwuoke et al. [41] proposed a division protocol to support encrypted floating point numbers based on homomorphic encryption. However, both of the above schemes use fixed rounds of iterative computations to guarantee fixed precise of results, which results in high computational overhead.

Secure Division Based on Secure Bit Decomposition Protocol (SBD)

The modulo value operation limits the length of the data in division computation. In order to protect the confidentiality of both the divisor and the dividend, some studies use the secure bit decomposition protocol [42] to realize secure division [6, 20]. 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.

Cybersecurity Education

With the development of cloud computing, the information system of organizations or schools becomes large-scale and complicated, which makes it difficult to deploy defensive mechanisms and suffers from undetected cyber-attacks [43]. Cybersecurity education aims to train IT-related employee or the future generation with technical skills. To customize specified cybersecurity exercises and enhance entities' security knowledge [44], a lot of data (such as work environment and threats) should be provided, which may breach privacy.

However, currently most researches [45–47] focus on the design and implementation of frameworks for cybersecurity exercises and testbeds. The STEAM framework [46] inserts the Arts into cybersecurity education, while the EDU Range framework [45] eliminates the dependence on virtual machine or private cloud replaced by public cloud-based framework. Frank et al. [47] introduced life cycle into testbed design. Abir et al. [48] pointed out that universities and industries lack communications about training courses and curriculum, which leads to that students do not gain adequate knowledges required by the specific workplace. To solve this issue, the cooperative education was designed to enhance the involvement of students and industry and enrich their work skills [49]. Rakesh et al. [50] discussed about the significance of security analytics and shared their educational experiences. But all work above ignored the privacy issues and do not provide a secure and privacy-preserving way to share data and customize courses.

3 System Model

Our proposed system mainly comprises five types of entities as shown in Fig. 1:

Data service provider (DSP) is served by the cloud, which stores user data, provides some computation service, and controls user access.

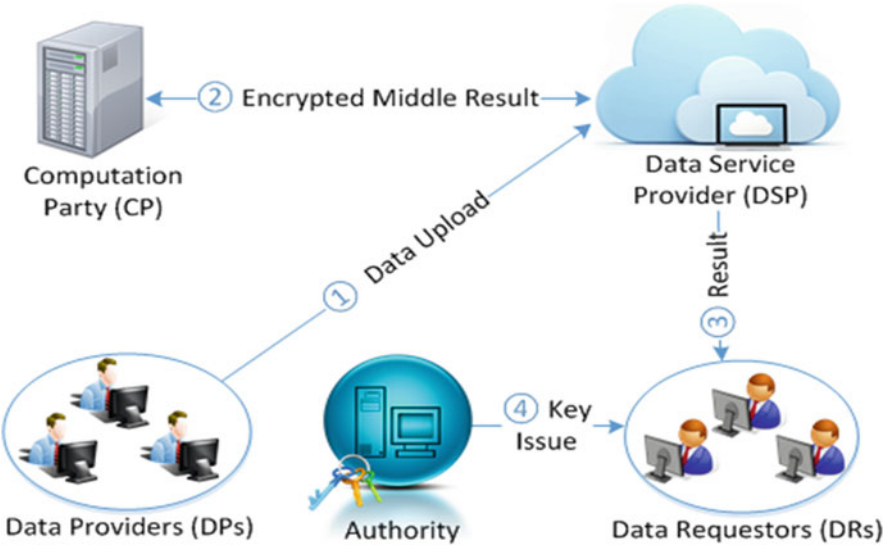

Fig. 1 A system model

Computation party (CP) fulfills partial computations and control access. It can be any party (a company or a university) who wants to train its employee or students with cybersecurity skills. There may exist multiple CPs for different applications. Herein, we simplify our design by considering only one CP in this chapter.

Data providers (DPs) are the data collectors or producers that encrypt data and store them in the DSP.

Data requesters (DRs) are the data consumers that acquire the result of data processing in a specific context.

Authority is fully trusted, which is responsible for system parameter generation and ABE key issue.

4 Preliminaries

For a better understanding of the scheme designs, please refer to previous work [51–53] for the detailed notation tables.

Additive Homomorphic Encryption

Paillier's cryptosystem [54] is one of the most important additive homomorphic encryption. Suppose we have N pieces of encrypted data under same key pk, which

can be presented as $[m_i]_{pk}$ $(i = 1, 2, \ldots, N)$. Additive homomorphic encryption satisfies this equation, $D_{sk}\left(\prod_{i=1}^{N}[m_i]_{pk}\right) = \sum_{i=1}^{N} m_i$, where $D_{sk}()$ is the homomorphic decryption algorithm with secret key *sk*.

Key-Policy Attribute-Based Encryption (KP-ABE)

In KP-ABE, ciphertexts are generated based on some descriptive attributes, while decryption keys are associated with policies. For more details about KP-ABE, refer to [32]. Notably, ciphertext-policy attribute-based encryption (CP-ABE) [31] can also be applied to implement our scheme.

Homomorphic Re-Encryption Scheme (HRES)

We revise the Scheme [55] (named as EDD) and design the HRES to provide two-level decryption and achieve secure data processing. The complete version of HRES is introduced in work [19].

Data Processing Procedure

Step 1 (System Setup @ All Entities): Authority calls the algorithm ***KeyGen*** and $Setup^{ABE}(\lambda, U)$ to complete the setup of HRES and ABE.

Step 2 (Data Upload @ DPs): DPs encrypt their personal data before uploading it to the DSP. It directly recalls ***EncTK*** to encrypt data m_i (unless otherwise specified, $|m_i| < \mathcal{L}(n)/4$):

$$[m_i] = \left(T_i, T_i'\right) = \left\{(1 + m_i * n) * PK^{r_i}, g^{r_i}\right\} \bmod n^2$$

Step 3 (Data Preparation @ DSP): Upon receiving the data from DPs, the DSP needs to do some analyses over the encrypted data. It provides a data packet and ABE ciphertext for access control to the CP. In addition, CP chooses a random partial key ck_1 for access control, which will be used in Step 5.

Step 4 (Data Process @ CP): Upon receiving the preprocessing results from DSP, CP chooses another random partial key ck_2 to obtain the preprocessing result $\left[\hat{m}\right]_{pk_{ck_2}}$ or $\left[\hat{f}\right]_{pk_{ck_2}}$. Regarding access control, CP encrypts ck_2 using ABE to get $CK_2' = Enc^{ABE}\left(ck_2, \gamma, PK'\right)$ and forwards it to DSP.

Step 5 (Additional Process @ DSP): The DSP needs to remove the mask from ciphertext $\left[\hat{m}\right]_{pk_{ck_2}}$ or $\left[\hat{f}\right]_{ck_2}$ to obtain processed ciphertext $[m]_{pk_{ck}}$ or $[f]_{pk_{ck}}$ where $pk_{ck} = g^{ck}$ and $ck = ck_1 * ck_2$.

Regarding access control, the DSP encrypts ck_1using ABE under the same policy to get CK'_1 and further gets CK' through the homomorphism of ABE:$CK' = CK'_1 * CK'_2 = Enc^{ABE}\left(ck_1 * ck_2, \gamma, PK'\right)$. Finally, the DSP keeps $[m]_{pk_{ck}}$ or $[f]_{pk_{ck}}$ and CK' for user access.

Step 6 (Data Access @ DR): If the DR satisfies the access policy, Authority issues a secret key SK' to the DR. Hence, the DR can decrypt CK' to get ck and further obtain m or f.

5 Detailed Data Processing

System setup and data collection are the same as those in part 4. Thus, we do not introduce the details in this part; we mainly focus on the steps from 3 to 6 in each basic operation.

Addition

This function aims to obtain the sum of all raw data, $m = \sum_{i=1}^{N} m_i$, which can be accomplished by multiplying all ciphertexts. Note that the number of the data in *Addition* affects the length of the provided data. If we want to get the sum result of N pieces of data, it should guarantee that $m_i < n/N$.

Step 3 (Data Preparation @ DSP): Due to additive homomorphism, the DSP can directly multiply encrypted data one by one as follows:$[m] = \left(T, T'\right) = \prod_{i=1}^{N}[m_i] = \left(\prod_{i=1}^{N} T_i, \prod_{i=1}^{N} T_i{}'\right)$. To realize group access control, it chooses a random number r_1 and the first partial key ck_1 and then computes as follows:

1. Compute $c_1 = {ck_1}^{-1}\ mod\ n^2$.
2. Mask ciphertext: $[c_1\ (m + r_1)] = \left(\widetilde{T}, \widetilde{T'}\right) = \left\{(T\ (1 + r_1 * n))^{c_1}, \left(T'\right)^{c_1}\right\}$.
3. Call ***PDec*1** to partially decrypt it:$[c_1\ (m + r_1)]_{pk_{CP}} = \left(\hat{T}, \hat{T'}\right) = \left\{\widetilde{T}, \left(\widetilde{T\prime}\right)^{a}\right\}$.

Then DSP sends $[c_1\ (m + r_1)]_{pk_{CP}}$ to the CP.

Step 4 (Data Process @ CP): The CP calls the algorithm ***PDec*2** with sk_{CP} to decrypt the encrypted data and obtain $c_1(m + r_1)$. Then the CP chooses the second partial key ck_2 and a random number r to encrypt data as follows, $[c_1\ (m + r_1)]_{pk_{ck_2}} = \left\{(1 + c_1\ (m + r_1)\ n)\ g^{ck_2 * r}, g^r\right\}$,where $pk_{ck_2} = g^{ck_2}$. The CP encrypts ck_2 to obtain CK'_2 and then forwards $\left[\hat{m}\right]_{pk_{ck}}$ and CK'_2 back to the DSP.

Step 5 (Additional Process @ DSP): The DSP computes to obtain the final processed data with ck_1 and r_1, $[m]_{pk_{ck}} = \left(\overline{T}^{ck_1}(1 - r_1 n), \overline{T}'\right) = \left\{(1 + m * n) g^{ck_1 * ck_2 * r}, g^r\right\}$, where $pk_{ck} = g^{ck_1 * ck_2}$ and $ck = ck_1 * ck_2$. It encrypts ck_1 using ABE and gets $CK' = CK'_1 * CK'_2 = Enc^{ABE}\left(ck_1 * ck_2, \gamma, PK'\right)$.

Subtraction

This function aims to obtain the subtraction of some data $\left(m = \sum_{i=1}^{W} m_i - \sum_{i=\mathrm{W}}^{N} m_i\right)$ with encrypted data $[m_i]$ $(i = 1, \ldots, N)$. It can be accomplished by negating the subtracted terms (by raising to the power of $(n - 1)$) and then following the procedure of *Addition*.

Step 3 (Data Preparation @ DSP): The DSP first computes $\left[\sum_{i=1}^{W} m_i\right] = \prod_{i=1}^{W} [m_i]$ and $\left[\sum_{i=W+1}^{N} m_i\right] = \prod_{i=W+1}^{N} [m_i]$. It further calculates $\left[-\sum_{i=W+1}^{N} m_i\right] = \left(\left[\sum_{i=W+1}^{N} m_i\right]\right)^{n-1}$ and multiplies them to obtain: $[m] = \left[\left(\sum_{i=1}^{W} m_i - \sum_{i=W+1}^{N} m_i\right)\right] = \left[\sum_{i=1}^{W} m_i\right] * \left[-\sum_{i=W+1}^{N} m_i\right]$. Then the subsequent process is the same to that in *Addition*. Due to length and simplicity reasons, we skip its details.

Multiplication

This function aims to obtain the product of all raw data $\left(m = \prod_{i=1}^{N} m_i\right)$. For ease of presentation, we describe the details with two pieces of data ($[m_1]$, $[m_2]$). Note that if we need to get the product of N pieces of data, it must be guaranteed that $\mathcal{L}(m_i) < \mathcal{L}(n)/(2N)$.

Step 3 (Data Preparation @ DSP): First, the DSP chooses a random partial key ck_1 and a random number c_1 and sets another one as $c_2 = (ck_1 * c_1)^{-1} \bmod n$.

To conceal each raw data from the CP, the DSP does one exponentiation and one decryption with its own secret key by calling ***PDec1***:

1. $[c_1 * m_1] = \left\{T_1{}^{c_1}, \left(T_1'\right)^{c_1}\right\}; [c_1 * m_1]_{pk_{CP}} = \left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T_1{}^{c_1}, \left(T_1'\right)^{c_1 * a}\right\}$
2. $[c_2 * m_2] = \left\{T_2{}^{c_2}, (T_2')^{c_2}\right\}; [c_2 * m_2]_{pk_{CP}} = \left(T_2{}^{(1)}, T_2{}'^{(1)}\right) = \left\{T_2{}^{c_2}, (T_2')^{c_2 * a}\right\}$.

The data packet sent to the CP is $\{[c_1 * m_1]_{pk_{CP}}, [c_2 * m_2]_{pk_{CP}}$.

Step 4 (Data Process @ CP): Upon receiving the data packet from the DSP, the CP uses the algorithm ***PDec2*** to decrypt the data: $c_1 * m_1 = T_1{}^{(1)}/(T_1{}'^{(1)})^b$, $c_2 * m_2 = T_2{}^{(1)}/(T_2{}'^{(1)})^b$.

It then chooses ck_2 and a random number r and encrypts $c_1 * m_1 * c_2 * m_2$ and ck_2 as follows:

$\left[\hat{m}\right]_{pk_{ck_2}} = [c_1c_2m]_{pk_{ck_2}} = \left\{(1 + c_1m_1c_2m_2 * n)\, g^{ck_2*r}, g^r\right\}$; $CK'_2 = Enc^{ABE}\left(ck_2, \gamma, PK'\right)$.

Finally, the CP forwards $\left[\hat{m}\right]_{pk_{ck_2}}$ and CK'_2 to the DSP.

Step 5 (Additional Process @ DSP): The DSP further processes the data packet with ck_1 and gets ciphertext as follows:$[m]_{pk_{ck}} = \left\{\overline{T}^{ck_1}, \overline{T}'\right\}$;$CK' = CK'_2 * Enc^{ABE}\left(ck_1, \gamma, PK'\right)$.

Sign Acquisition

We assume that $\mathcal{L}(m) < \mathcal{L}(n)/4$ and *BIG* is the largest raw data of m. Then the raw data is in the scope $[-BIG, BIG]$. Sign Acquisition can be achieved by masking the original ciphertext with random numbers of limited length and then checking the length of the masked data to further determine the real length of original data. Here, the DR targets to obtain the final sign indicator f from $[m_1]$.

Step 3 (Data Preparation @ DSP): The DSP chooses three random numbers R $\left(\mathcal{L}(R) < \mathcal{L}(n)/4\right)$, c_1, and ck_1. It first encrypts "1" and then computes as follows:

1. $[1] = \{(1 + n) * PK^{r'}, g^{r'}\}$;$[2 * m_1 + 1] = (T, T') = [m_1]^2 * [1]$.
2. Then it flips a coin s. If $s = -1$, it computes:

$$\left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T^{n-R}, \left(T'\right)^{a*(n-R)}\right\} = [-R * (2 * m_1 + 1)]$$

Otherwise, if $(s = 1)$, it calls ***PDec*1** and computes:

$$\left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T^R, T'^{a*R}\right\} = [R * (2 * m_1 + 1)]$$

3. The DSP Computes $c_2 = (ck_1)^{-1} \bmod n$ and $s' = c_1 * c_2 * s \bmod n$.

The data packet sent to the CP is $\{(T_1{}^{(1)}, T_1{}'^{(1)}), s'\}$.

Step 4 (Data Process @ CP): Upon receiving the data packet from the DSP, the CP decrypts $(T_1{}^{(1)}, T_1{}'^{(1)})$ with ***PDec*2** to obtain raw data $m' = R * (2 * m_1 + 1) \bmod n$ if $s = 1$ or $m' = R * (2 * m_1 + 1) \bmod n$ if $s = -1$. The CP compares $\mathcal{L}\left(m'\right)$ with $\mathcal{L}(n)/2$. If $\mathcal{L}\left(m'\right) < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$.

The CP chooses a random number r and a second partial key ck_2 and further computes as follows:$\left[\hat{f}\right]_{pk_{ck_2}} = \left(\overline{T}, \overline{T}'\right) = \left\{\left(1 + s'u * n\right) g^{ck_2*r}, g^r\right\}$. Encrypt ck_2 using ABE: $CK'_2 = Enc^{ABE}\left(ck_2, \gamma, PK'\right)$.

Finally, the CP forwards $\left[\hat{f}\right]_{pk_{ck_2}}$ to DSP.

Step 5 (Additional Process @ DSP): The DSP further processes the data packet as follows:

Compute $c_3 = {c_1}^{-1} \bmod n$;$[f]_{pk_{ck}} = \left\{\overline{T}^{ck_1 * c_3}, \left(\overline{T}'\right)^{c_3}\right\}$;$CK' = Enc^{ABE}(ck_1, \gamma, PK') * CK'_2$.

Step 6 (Data Access @ DR): The DR satisfying the access policy in ABE can decrypt $CK^{'}$ to obtain ck and further decrypts $[f]_{pk_{ck}}$ to obtain f. Note: if $f = 1$, $m_1 \geq 0$; otherwise, $m_1 < 0$.

Absolute

We assume that $\mathcal{L}(m) < \mathcal{L}(n)/4$ and that *BIG* is the largest raw data of m. Then the raw data is in the scope $[-BIG, BIG]$. Here, given ciphertext $[m_1]$, DR wants to get the absolute value $|m_1|$.

Step 3 (Data Preparation @ DSP): The DSP chooses three random numbers R where $\mathcal{L}(R) < \mathcal{L}(n)/4$, c_1, and c_2 and chooses the first partial key ck_1. It first encrypts "1" and computes as follows:

1. $[1] = \{(1+n) * PK^{r'}, g^{r'}\}$;$[2 * m_1 + 1] = (T, T') = [m_1]^2 * [1]$.
2. Then it flips a coin s. If $s = -1$, $({T_1}^{(1)}, {T_1}'^{(1)}) = [-R * (2 * m_1 + 1)]$.
   Otherwise, it calls ***PDec1*** and computes $({T_1}^{(1)}, {T_1}'^{(1)}) = [R * (2 * m_1 + 1)]$.
3. Compute $[c_1 m_1] = [m_1]^{c_1}$, and call ***PDec1*** to obtain $[c_1 m_1]_{pk_{CP}}$.
4. The DSP sets $c_3 = (ck_1)^{-1} \bmod n$ and $s^{'} = c_2 * c_3 * s \bmod n$.

The data packet sent to the CP is $\{\left({T_1}^{(1)}, {T_1}'^{(1)}\right), s', [c_1 m_1]_{pk_{CP}}\}$.

Step 4 (Data Process @ CP): Upon receiving the data packet from DSP, the CP decrypts $({T_1}^{(1)}, {T_1}'^{(1)})$ and $[c_1 m_1]_{pk_{CP}}$ with ***PDec2*** to obtain raw data: $m^{'} = (-1)^{s+1} * R * (2 * m_1 + 1) \ mod \ n$ and $c_1 m_1$, respectively. CP compares $\mathcal{L}(m')$ with $\mathcal{L}(n)/2$. If $\mathcal{L}(m') < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$. Then CP chooses r and the second partial key ck_2 and further computes as follows:$\left[c_1 m_1 s' u\right]_{pk_{ck_2}} = \left(\overline{T}, \overline{T}'\right)$. Encrypt c$k_2$with ABE: $CK'_2 = Enc^{ABE}\left(ck_2, \gamma, PK'\right)$. Finally, the CP forwards $\left[c_1 m_1 s' u\right]_{pk_{ck_2}}$ and CK'_2 to DSP.

Step 5 (Additional Process @ DSP): The DSP further processes the data packet as follows:

1. Set $c_4 = (c_1)^{-1} \ mod \ n$ and $c_5 = (c_2)^{-1} \bmod n$.

$$[su * m_1]_{pk_{ck}} = \left\{\overline{T}^{ck_1 * c_4 * c_5}, \overline{T}'^{c_4 * c_5}\right\}; CK' = Enc^{ABE}\left(ck_1, \gamma, PK'\right) * CK'_2$$

Step 6 (Data Access @ DR): The DR that satisfies the access policy in ABE can decrypt $CK^{'}$ to obtain ck. The DSP sends the data packet $[su * m_1]_{pk_{ck}}$ to the DR in a secure way. Then the DR can decrypt it to obtain $su * m_1$. Note: if $m_1 \geq 0$, $su = 1$; otherwise, $su = -1$. Hence, $su * m$ is the absolute of data m.

Comparison

Comparison can be simply accomplished by checking the sign of the difference value of two data by calling *Sign Acquisition*. For ease of presentation, $m_1 - m_2$ is denoted as m_{1-2}.

$[m_1] = \left(T_1, T_1{}'\right) = \{(1 + m_1 * n) * PK^{r_1}, g^{r_1}\}; [m_2] = \left(T_2, T_2{}'\right) = \{(1 + m_2 * n) * PK^{r_2}, g^{r_2}\}$

Step 3 (Data Preparation @ DSP): DSP first computes to get the subtraction of encrypted data:

$$\left(T, T'\right) = \left\{T_1 * (T_2)^{n-1}, T_1{}' * (T_2{}')^{n-1}\right\} = [(m_1 - m_2)].$$

The following steps are the same as those in *Sign Acquisition*, which are skipped for the reason of chapter length limitation. Through the cooperation of the DSP and the CP, the DR finally gets the sign of $m_{1-2} = m_1 - m_2$. DR can obtain the comparison result. If $m_{1-2} \geq 0$, $m_1 \geq m_2$; otherwise, $m_1 < m_2$.

Equality Test

Equality test needs to check the signs of both difference value and negative difference value of original two data by calling *Comparison* twice. DR wants to know whether m_1 is equal to m_2 or not from encrypted data ($[m_1]$, $[m_2]$). The DSP and CP directly interact with each other in two parallel computations of *Comparison*.

They compare m_1 and m_2 in two forms: 1) $m_{1-2} = m_1 - m_2$ and 2) $m_{2-1} = m_2 - m_1$. Through the operations in *Comparison*, DSP can get two results $[f_1]_{pk_{ck}}$ and $[f_2]_{pk_{ck}}$, respectively. Then the DSP can obtain $[f]_{pk_{ck}} = [f_1 + f_2]_{pk_{ck}} = [f_1]_{pk_{ck}} * [f_2]_{pk_{ck}}$. Finally, DR that satisfies the access policy in ABE can decrypt CK' to obtain ck. DSP sends the data packet $[f]_{pk_{ck}}$ to the DR in a secure way. Then the DR can further decrypt $[f]_{pk_{ck}}$ to obtain f. Note: if $f = 2$, $m_1 = m_2$; otherwise, $m_1 \neq m_2$.

Maximum and Minimum

Two-to-One (T2O)

This scheme aims to obtain the max and min values from two encrypted data for a data requester.

Step 3 (@ DSP): First, the DSP randomly selects some numbers R_1, R_2, and $R_3$where $\mathcal{L}(R_1) < \mathcal{L}(n)/4$ and then executes the following operations: here, $m\_ = m_1 - m_2$ and $m_+ = m_1 + m_2$.

1. $[1] = \{(1+n) * PK^{r'}, g^{r'}\}; [m_-] = [m_1 - m_2] = [m_1] * [m_2]^{n-1}$
2. $[R_2 * m_+ + R_3] = ([m_1 + m_2])^{R_2} * [R_3]; [R_2 m_-] = \left(T_-, T'_-\right) = [m_1 - m_2]^{R_2}$

$$[2 * m_- + 1] = \left(T, T'\right) = \left\{(1 + (2 * m_- + 1) * n) * PK^{r'+2*r_1}, g^{r'+2*r_1}\right\}$$

Then it flips a coin s. If $s = -1$, then compute$\left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T^{n-R_1}, \left(T'\right)^{a*(n-R_1)}\right\} = [-R_1 * (2 * m_- + 1)]_{pk_{CP}}$ and$\left(T_2, T'_2\right) = [-R_2 m_-]_{pk_{CP}} = \left\{T_-{}^{n-R_1}, \left(T'_-\right)^{a*(n-R_1)}\right\}$. Otherwise, if $(s = 1)$, it calls $PDec1$ and computes $\left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T^{R_1}, T'^{a*R_1}\right\} = [R_1 * (2 * m_- + 1)]_{pk_{CP}}$ and $\left(T_2, T'_2\right) = [R_2 m_-]_{pk_{CP}} = \left\{T_-, \left(T'_-\right)^a\right\}$. It further calls $PDec1$ on $[R_2 * m_+ + R_3]$ to get $[R_2 * m_+ + R_3]_{pk_{CP}}$. Finally, it forwards CP the data packet $\{\left(T_1{}^{(1)}, T_1{}'^{(1)}\right), [R_2 * m_+ + R_3]_{pk_{CP}}, \left(T_2, T'_2\right)\}$.

Step 4 (@ CP): CP further processes the data packet from the DSP. It first decrypts $(T_1{}^{(1)}, T_1{}'^{(1)})$ and $\left(T_2, T'_2\right)$with $PDec2$ to obtain raw data $\hat{m} = R_1 * (2 * m_- + 1) \bmod n$,$\hat{m_-} = (R_2 m_-) \bmod n$ if $s = 1$ or $\hat{m} = -R_1 * (2 * m_- + 1) \bmod n$,$\hat{m_-} = (-R_2 m_-) \bmod n$ if $s = -1$.

Then CP needs to compare $\mathcal{L}\left(\hat{m}\right)$ with $\mathcal{L}(n)/2$. If $\mathcal{L}\left(\hat{m}\right) < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$. The CP further encrypts the raw data $u * \hat{m_-}$ with the public key of the targeted DR as $\left[u * \hat{m_-}\right]_{pk_{DR}} = \left(\overline{T}, \overline{T}'\right) = \left\{\left(1 + u\hat{m_-} * n\right) pk_{DR}{}^r, g^r\right\}$.

Decrypt $[R_2 * m_+ + R_3]_{pk_{CP}}$ and then encrypt it with pk_{DR} to get $[R_2 * m_+ + R_3]_{pk_{DR}}$. Finally, the CP forwards the data packet to DSP: $\{\left[u * \hat{m_-}\right]_{pk_{DR}}, [R_2 * m_+ + R_3]_{pk_{DR}}\}$.

Step 5 (@ DSP): The DSP first removes the mask R_3 by computing $[R_2 m_+]_{pk_{DR}} = [R_2 * m_+ + R_3]_{pk_{DR}} * [-R_3]_{pk_{DR}}$. Then it can get the max and min with $r = (2R_2)^{-1} \bmod n$:

$$[\max]_{pk_{DR}} = \left(\left[u * \hat{m_-}\right]_{pk_{DR}} * [R_2 m_+]_{pk_{DR}}\right)^r;$$

$$[\min]_{pk_{DR}} = \left(\left[u * \hat{m_-}\right]_{pk_{DR}}{}^{n-1} * [R_2 m_+]_{pk_{DR}}\right)^r.$$

Step 6 (@ DR): The DR with the corresponding secret key can decrypt the ciphertext ($[\max]_{pk_{DR}}$ and $[\min]_{pk_{DR}}$) to obtain the maximum and minimum values.

Multiple-to-One (M2O)

Given an example of n pieces of ciphertexts ($[m_1]$,$[m_2]$,$\cdots [m_i]$,$\cdots [m_n]$), this scheme can get the maximum and minimum results $[\max]_{pk_{DR}}$ and $[\min]_{pk_{DR}}$ for the targeted data requester DR. Note that the T2O can provide the maximum and minimum values from ciphertext $[m_1]$ and $[m_2]$ for DR. If we use the PK to

replace the public key of DR (pk_{DR}) in T2O, we can get the ciphertext [*max*] and [*min*] through parallel processing. Herein, we take maximum computation as an example, which has the same procedure as minimum computation.

In order to get the final maximum from more than two ciphertext, we need to execute several rounds of the T2O scheme. The computation follows a tree structure. It divides the data into many groups and each group has two pieces of data. Then T2O is executed over every group with *PK* to get the ciphertext [*max*]. Until the last two pieces of data in the last layer, DSP and CP execute T2O with pk_{DR} to get the final ciphertext $[\max]_{pk_{DR}}$.

Two-to-Multiple (T2M)

Given two ciphertext $[m_1]$ and $[m_2]$, this scheme can provide the sorting results $[\max]_{pk_{ck}}$ and $[\min]_{pk_{ck}}$, which indicates the ciphertext of max and the min results under the public key pk_{ck}.

Step 3 (@ DSP): DSP randomly selects four numbers, R_1, R_2, R_3, ck_1, which satisfies $R_1 = R_2 * ck_1 \bmod n^2$ and $\mathcal{L}(R_1) < \mathcal{L}(n)/4$ and then preprocesses the data from DPs as follows:

1. $[1] = \{(1+n) * PK^{r'}, g^{r'}\}; [m_-] = [m_1 - m_2] = [m_1] * [m_2]^{n-1}$
2. $[R_2 m_+ + R_3] = [m_1 + m_2]^{R_2} * [R_3]; [R_2 m_-] = \left(T_-, T'_-\right) = [m_1 - m_2]^{R_2}$

$$[2 * m_- + 1] = \left(T, T'\right) = \left\{(1 + (2 * m_- + 1) * n) * PK^{r'+2*r_1}, g^{r'+2*r_1}\right\}$$

The DSP calls ***PDec*1** to decrypt $[R_2 * m_+ + R_3]$ to get $[R_2 * m_+ + R_3]_{pk_{CP}}$. Then, it further flips a coin *s*. If $s = -1$, it computes $\left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T^{n-R_1}, \left(T'\right)^{a*(n-R_1)}\right\} = [-R_1 * (2 * m_- + 1)]_{pk_{CP}}$, $\left(T_2, T'_2\right) = [-R_2 m_-]_{pk_{CP}} = \left\{T_-{}^{n-R_1}, \left(T'_-\right)^{a*(n-R_1)}\right\}$. Otherwise if $(s = 1)$, it directly calls ***PDec*1** to compute $\left(T_1{}^{(1)}, T_1{}'^{(1)}\right) = \left\{T^{R_1}, T'^{a*R_1}\right\} = [R_1 * (2 * m_- + 1)]_{pk_{CP}}$, $\left(T_2, T'_2\right) = [R_2 m_-]_{pk_{CP}} = \left\{T_-, \left(T'_-\right)^a\right\}$. Then it sends CP the data packet $\{\left(T_1{}^{(1)}, T_1{}'^{(1)}\right), \left(T_2, T'_2\right), [R_2 * m_+ + R_3]_{pk_{CP}}\}$.

Step 4 (@ CP): The CP calls *PDec*2 to decrypt $(T_1{}^{(1)}, T_1{}'^{(1)})$ and $\left(T_2, T'_2\right)$from DSP to obtain raw data $m' = R_1 * (2 * m_- + 1) \ mod \ n, \hat{m_-} = (R_2 m_-) \bmod n$ if s = 1 or $m' = -R_1 * (2 * m_- + 1) \ mod \ n, \hat{m_-} = (-R_2 m_-) \bmod n$ if s = −1.

The CP checks the sign of m' by comparing $\mathcal{L}\left(m'\right)$ with $\mathcal{L}(n)/2$. If $\mathcal{L}\left(m'\right) < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$. And it further encrypts the raw data $u * \hat{m_-}$ with a randomly chosen key pair $(ck_2, pk_{ck_2} = g^{ck_2})$: $\left[u * \hat{m_-}\right]_{pk_{ck_2}} = \left(\overline{T}, \overline{T}'\right) = \left\{\left(1 + u\hat{m_-} * n\right) g^{ck_2 * r}, g^r\right\}$.

Decrypt $[R_2 * m_+ + R_3]_{pk_{CP}}$ to get $R_2 * m_+ + R_3$ and re-encrypt it as $[R_2 * m_+ + R_3]_{pk_{ck_2}}$. Moreover, it needs to encrypt ck_2 with ABE to get

$CK'_1 = Enc^{ABE}\left(ck_2, \gamma, PK'\right)$. Finally, the CP forwards the data packet to DSP: $\{\left[u * \hat{m_-}\right]_{pk_{ck_2}}, [R_2 * m_+ + R_3]_{pk_{ck_2}}, CK'_1\}$.

Step 5 (@ DSP): First, the DSP sets $\{T,\ T^{'}\}=[R_2 * m_+ + R_3]_{pk_{ck_2}}$ and then computes $[R_2 * m_+]_{pk_{ck_2}} = \left\{T * (1 - R_3 * n), T'\right\}$. The DSP computes $r = (2R_1)^{-1}\ mod\ n$ and finally obtains the encrypted max and min: $[\max]_{pk_{ck}} = \left(\left(\left[u * \hat{m_-}\right]_{pk_{ck_2}} * [R_2 * m_+]_{pk_{ck_2}}\right)^{1,ck_1}\right)^r$; $[\min]_{pk_{ck}} = \left(\left(\left(\left[u * \hat{m_-}\right]_{pk_{ck_2}}\right)^{n-1} * [R_2 * m_+]_{pk_{ck_2}}\right)^{1,ck_1}\right)^r$.

The DSP calls HE^{ABE} to obtain $CK = CK'_1 * Enc^{ABE}\left(ck_2, \gamma, PK'\right) = Enc^{ABE}\left(ck_1 * ck_2, \gamma, PK'\right)$.

Step 6 (@ DR): The DR can access the computation results if it satisfies the access policy.

Multiple-to-Multiple (M2M)

DSP and CP invoke the T2M rather than the T2O to obtain the final result $\left[\max_{\lfloor lb(n) \rfloor, 1}\right]_{pk_{ck}}$. Owing to chapter length limitation, we skip the details of above process.

Division

Scheme 1

Scheme 1 can provide the ciphertext of division result $[\lfloor m_1/m_2 \rfloor]_{pk_{DR}}$ as shown in Fig. 2.

Step 3 (Data Preparation @ DSP): DSP first chooses two random numbers r_1, r_2, where $L(r_i) < L(n)/4$. Then, it processes data to conceal each raw data from CP, as described below:

1. $[m_1 r_1] = \left\{T_1{}^{r_1}, \left(T'_1\right)^{r_1}\right\} = [m_1]^{r_1}, [m_2 r_1] = \left\{T_2{}^{r_1}, \left(T'_2\right)^{r_1}\right\} = [m_2]^{r_1}$.
2. $[m_2 r_1 r_2] = [m_2 r_1]^{r_2} = [m_2]^{r_1 r_2} = \left\{T_2{}^{r_1 r_2}, \left(T'_2\right)^{r_1 r_2}\right\}$
   ; $[m_1 r_1 + m_2 r_1 r_2] = [m_1 r_1]_* [m_2 r_1 r_2]$.
3. $[m_2 r_1]_{pk_{CP}} = \left\{T_2{}^{r_1}, \left(T'_2\right)^{r_1 * sk_{DSP}}\right\} = \left\{(1 + r_1 * m_2 * n)\ PK^{r*r_1}, g^{r*a*r_1}\right\}$.

$$[m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}} = \left\{T_1{}^{r_1} T_2{}^{r_1 r_2}, \left(T'_2\right)^{r_1}\left(T'_2\right)^{r_1 r_2}\right\}^a.$$

Next, DSP sends the data packet $\left([m_2 r_1]_{pk_{CP}}, [m_1 r_1 + m_2 r_1 r_2]_{pk_{CP}}\right)$ to CP.

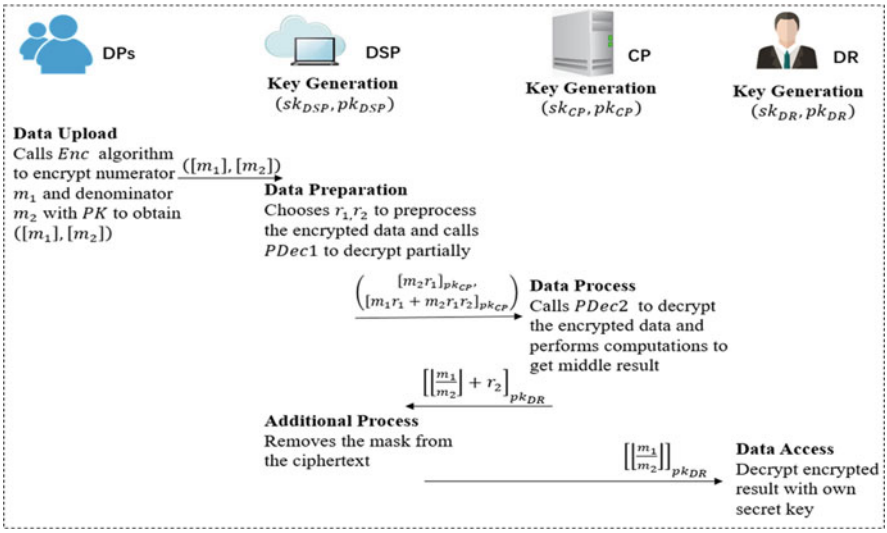

Fig. 2 The procedure of division computation for a targeted data requester

Step 4 (Data Process @ CP): Upon receiving the data packet from DSP, CP calls ***PDec2*** to decrypt the packet. Then, CP performs division operations on plaintexts and encrypts the computational result with pk_{DR}.

1. $m_2r_1 = {T_2}^{r_1} / \left(\left(T_2'\right)^{r_1 * a}\right)^b \bmod n; m_1r_1 + m_2r_1r_2$
   $= {T_1}^{r_1} {T_2}^{r_1 r_2} / \left(\left(T_2'\right)^{r_1} \left(T_2'\right)^{r_1 r_2}\right)^{a*b} \bmod n.$
2. $\left(m_1r_1 + m_2r_1r_2\right) / m_2r_1 = \left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2; \left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{DR}}$
   $= \left\{\left(1 + \left(\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right) * n\right) pk_{DR}^r, g^r\right\}.$

The data sent to DSP is the ciphertext $\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{DR}}$. We use $\left\lfloor \frac{m_1}{m_2} \right\rfloor$ to represent the quotient.

Step 5 (Additional Process @ DSP): DSP encrypts the random number r_2 as $[r_2]_{pk_{DR}}$ and computes $\left([r_2]_{pk_{DR}}\right)^{n-1}$. Then, DSP removes the mask from the ciphertext as below.

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

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

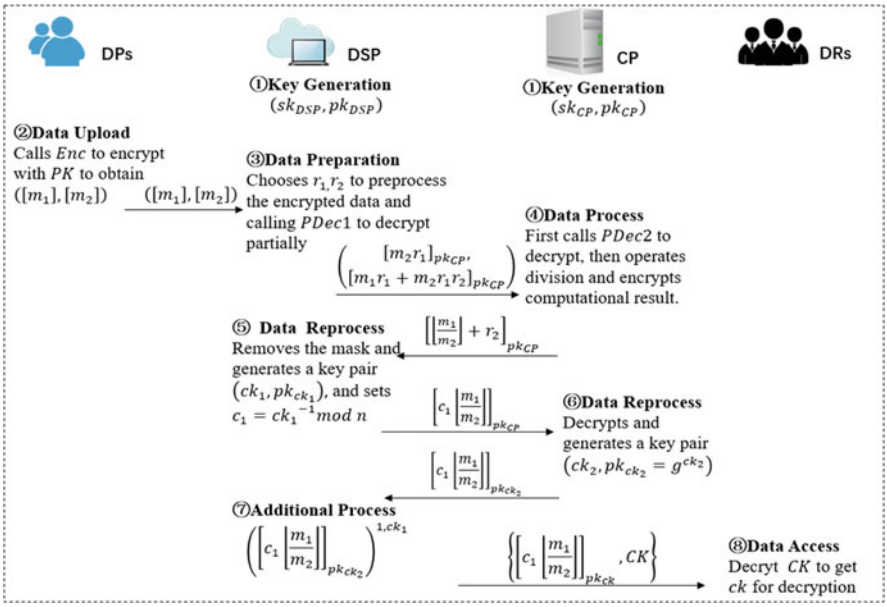

Fig. 3 The procedure of division computation with flexible access control

Scheme 2

We design Scheme 2 to enable flexible access control over computational results as shown in Fig. 3.

Step 3 (Data Preparation @ DSP): DSP chooses two random numbers r_1 and r_2 where $L(r_i) < L(n)/4$ and preprocesses data to mask raw data as follows, which is the same as Scheme 1.

1. $[m_1r_1] = \left\{T_1^{r_1}, \left(T_1'\right)^{r_1}\right\} = [m_1]^{r_1}, [m_2r_1] = \left\{T_2^{r_1}, \left(T_2'\right)^{r_1}\right\} = [m_2]^{r_1}$.
2. $[m_2r_1r_2] = [m_2]^{r_1r_2} = \left\{T_2^{r_1r_2}, \left(T_2'\right)^{r_1r_2}\right\}; [m_1r_1 + m_2r_1r_2] = [m_1r_1] * [m_2r_1r_2]$.
3. $[m_2r_1]_{pk_{CP}} = \left\{T_2^{r_1}, \left(T_2'\right)^{r_1 * sk_{DSP}}\right\} = \left\{(1 + r_1 * m_2 * n)\ PK^{r*r_1}, g^{r*a*r_1}\right\}$.

$$[m_1r_1 + m_2r_1r_2]_{PK_{CP}} = \left\{T_1^{r_1} T_2^{r_1r_2}, \left(T_2'\right)^{r_1} \left(T_2'\right)^{r_1r_2}\right\}^a.$$

Similarly, DSP sends the data packet $\left([m_2r_1]_{pk_{CP}}, [m_1r_1 + m_2r_1r_2]_{pk_{CP}}\right)$ to CP.

Step 4 (Data Process @ CP): CP calls ***PDec2***$(*,$ ***sk$_{CP}$***$)$ to decrypt received data from DSP to get m_2r_1 and $m_1r_1 + m_2r_1r_2$, and then performs division operations on plaintexts with perturbations, as well as encrypts the computational result by calling $Enc(*, pk_{CP})$.

1. $\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2 = \left(m_1 r_1 + m_2 r_1 r_2\right) / m_2 r_1$, where $\left\lfloor \frac{m_1}{m_2} \right\rfloor$ is quotient and remainder is ignored.
2. CP sends the data $\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{CP}} = \left\{\left(1 + \left(\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right) * n\right) pk_{CP}^r, g^r\right\}$ to DSP.

Step 5 (Data Reprocess @ DSP): DSP chooses a partial key ck_1 and sets a random number as $c_1 = (ck_1)^{-1}$ *mod n*. DSP removes the mask from the ciphertext and performs the following computations:

$\left[\left|\frac{m_1}{m_2}\right| + r_2\right]_{pk_{CP}} * \left([r_2]_{pk_{CP}}\right)^{n-1} = \left[\left|\frac{m_1}{m_2}\right|\right]_{pk_{CP}}$; $\left[c_1 \left|\frac{m_1}{m_2}\right|\right]_{pk_{CP}} = \left(\left[\left|\frac{m_1}{m_2}\right|\right]_{pk_{CP}}\right)^{c_1}$
$= \left\{\tilde{T}, \tilde{T}'\right\}$

The data sent to CP is $\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}}$.

Step 6 (Data Reprocess @ CP): CP first calls ***PDec2***$(*, \boldsymbol{sk_{CP}})$ to decrypt the received data. Then, it chooses a partial key ck_2 to generate a key pair $\left(ck_2, pk_{ck_2} = g^{ck_2}\right)$ and calls $Enc\left(*, pk_{ck_2}\right)$ to encrypt the data:$c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor = \widetilde{T} / \left(\widetilde{T}'\right)^b \bmod n$;$\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck_2}} = \left\{\left(1 + \left(c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right) * n\right) pk_{ck_2}^r, g^r\right\} = \left\{\overline{T}, \overline{T}'\right\}$.

In addition, CP calls Enc^{ABE} to encrypt ck_2:$CK_2 = Enc^{ABE}\left(ck_2, \mathcal{T}, PK'\right)$. Furthermore, the ABE key CK_2 is sent to DSP along with the ciphertext $\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck_2}}$.

Step 7 (Additional Process @ DSP): DSP operates partial modular computation on received ciphertext with its partial key ck_1 and performs ABE algorithms to obtain encrypted access keys.

$$\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck}} = \left(\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck_2}}\right)^{1, ck_1} = \left\{\overline{T}^{ck_1}, \overline{T}'\right\}$$

$$= \left\{\left(1 + ck_1 * c_1 * \left\lfloor \frac{m_1}{m_2} \right\rfloor * n\right) g^{ck_1 * ck_2 * r}, g^r\right\} = \left\{\left(1 + \left\lfloor \frac{m_1}{m_2} \right\rfloor * n\right) g^{ck * r}, g^r\right\},$$

where $pk_{ck} = \left(pk_{ck_2}\right)^{ck_1} = \left(pk_{ck_1}\right)^{ck_2}$.

1. Calling Enc^{ABE} to encrypt ck_1: $CK_1 = Enc^{ABE}\left(ck_2, \mathcal{T}, PK'\right)$.
2. ABE homomorphic computation: $CK = CK_1 * CK_2 = Enc^{ABE}\left(ck_1 * ck_2, \mathcal{T}, PK'\right)$.

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

Step 8 (Data Access @ DRs): Upon receiving the computational results and CK from DSP, the DRs who satisfy the access policy can obtain a secret key SK' from

the authority. Thus, the DRs can decrypt CK to get ck by calling Dec^{ABE} and get the final quotient by calling $Dec\left(\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck}}, ck\right)$.

Division and Rest

Scheme 3

To support accurate division computation, we design Scheme 3 to further calculate remainder based on Scheme 1. We omit the same first three steps as in Scheme 1 and introduce the additional part as below.

Step 4 (Data Process @ CP): Upon receiving the data packet from DSP, CP first calls $\boldsymbol{PDec2}(*, \boldsymbol{sk_{CP}})$ to obtain masked plaintext and performs the following computations:

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

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

Step 5 (Data Additional Process @ DSP): DSP removes the mask from received ciphertext to get encrypted quotient and remainder as follows:

$\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{DR}} = \left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{DR}} * \left([r_2]_{pk_{DR}}\right)^{n-1}; [R]_{pk_{DR}} = \left([Rr_1]_{pk_{DR}}\right)^{r_1{}^{-1}}$.

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

Scheme 4

Similarly, Scheme 4 is proposed by adding the computations of remainder based on Scheme 2. We introduce its details below by omitting the same first three steps as in Scheme 2.

Step 4 (Data Process @ CP): Upon receiving data packet from DSP, CP first calls $\boldsymbol{PDec2}(*, sk_{CP})$ to obtain two messages $m_2 r_1$ and $(m_1 r_1 + m_2 r_1 r_2)$. Then, it performs basic computations to get $\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2$ and Rr_1. Furthermore, CP calls $Enc(*, pk_{CP})$ to encrypt the computational result and sends the encrypted data packet $\left\{\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{CP}}, [Rr_1]_{pk_{CP}}\right\}$ to DSP.

Step 5 (Data Reprocess @ DSP): DSP first chooses a partial key ck_1 and sets a random number as $c_1 = (ck_1)^{-1}\ mod\ n$. Then, it removes the mask from received ciphertext and conceals the data by performing the following computations:

1. $\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor + r_2\right]_{pk_{CP}} * \left([r_2]_{pk_{CP}}\right)^{n-1} = \left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}}; \left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}} = \left(\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}}\right)^{c_1}$.
2. $\left([Rr_1]_{pk_{CP}}\right)^{r_1^{-1}} = [R]_{pk_{CP}}; [c_1 R]_{pk_{CP}} = \left([R]_{pk_{CP}}\right)^{c_1} = \left\{\hat{T}, \hat{T}'\right\}$.

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

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

$$\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{CP}} \overset{\boldsymbol{PDec2}(*, sk_{CP})}{\rightarrow} c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor \overset{Enc\left(*, pk_{ck_2}\right)}{\rightarrow} \left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck_2}}.$$

$$[c_1 R]_{pk_{CP}} \overset{\boldsymbol{PDec2}(*, sk_{CP})}{\rightarrow} c_1 R \overset{Enc\left(*, pk_{ck_2}\right)}{\rightarrow} [c_1 R]_{pk_{ck_2}}.$$

In addition, CP calls ABE encryption algorithm to encrypt ck_2:$CK_2 = Enc^{ABE}\left(ck_2, \mathcal{T}, PK'\right)$.

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

Step 7 (Additional Process @ DSP): Upon receiving the data packet, DSP performs the following operations:

1. $\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck}} = \left(\left[c_1 \left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck_2}}\right)^{1, ck_1}; [R]_{pk_{ck}} = \left([c_1 R]_{pk_{ck_2}}\right)^{1, ck_1}$.
2. Using Enc^{ABE} to encrypt ck_1: $CK_1 = Enc^{ABE}\left(ck_2, \mathcal{T}, PK'\right)$.
3. Homomorphism of ABE: $CK = CK_1 * CK_2 = Enc^{ABE}\left(ck_1 * ck_2, \mathcal{T}, PK'\right)$.

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

Step 8 (Data Access @ DR): The DRs that satisfy the access policy can obtain a secret key SK' from the authority, which can be used to get ck by calling $Dec^{ABE}(PK', SK', CK)$. Then DRs decrypt the received ciphertext $\left[\left\lfloor \frac{m_1}{m_2} \right\rfloor\right]_{pk_{ck}}$ and $[R]_{pk_{ck}}$ obtained from DSP to get the quotient and remainder.

6 Applications in Cybersecurity Education

Privacy-preserving data processing with ABE guarantees data security and user privacy. In the field of cybersecurity education, privacy-sensitive data are generated and issued, e.g., course feedback, survey inputs, security-related data for intrusion/malware detection provided by different parties for course exercises, multi-party sensitive data processing, etc. By analyzing these data in a privacy-preserving way, we can judge teaching performance, support further course improvement, offer essential course practice to allow students to deeply understand cybersecurity theories and technologies, etc. Herein, our schemes offer an efficient and privacy-preserving measure to conduct data analysis, which provides a good practice for students to understand homophonic encryption and its usage. Some concrete examples are listed below:

Privacy-Preserving Data Analysis

The feedbacks and opinions of all students and faculties are essential to improve course quality. Our schemes can be adopted for data collection and dispel privacy concerns. It can be used in the following two scenarios:

Teaching Performance Evaluation: Our schemes can collect, process, and analyze the student ratings in a privacy-preserving way, especially in online courses. With our schemes, students are encouraged to provide their feedback or survey inputs honestly. Furthermore, the students can select preferred courses by comparing different course evaluation results and personal study expectation.

Preparing and Rating Examination Questions: The design goal of flexible data sharing and access control in our schemes would be a key point for remote cooperation among experts or teachers.

Teachers can prepare examination questions cooperatively in a privacy-preserving and flexible way. Our schemes can protect the content of examination papers and enable the teachers to get the feedback of other teachers to assess the rationality of papers. Moreover, they can also be applied to exchange the statistics of examination results from students and complete remote rating through cooperation. This kind of online cooperation can greatly improve education efficiency.

Cybersecurity Experimental Platform

Apart from the above, our schemes can be integrated to build up an experimental platform for cybersecurity education. It will help students gain a deep insight into privacy and security of outsourced data processing.

Cybersecurity Course Exercises: Our schemes offer a good experimental platform to conduct cybersecurity experiments with regard to secure data analytics for flexible and fine-grained access control over the processing results. For example, a number of students can collect sensitive security-related data from different sources and perform secure processing on those data at an untrusted party, and then different students get the processing results without knowing other inputs. For another example, students can provide their own mobile phone apps' usage data to process in a secure way with our schemes in order to know the trust and popularity of the apps without disclosing their personal app usage information. Through these experimental exercises, the students can get deep insight on encrypted data processing and flexible access control over processing results.

7 Conclusion

With the development and widely deployment of information systems, cybersecurity education becomes popular and significant. In order to gain customized courses, some private information are offered but may erode their privacy. In this chapter, we proposed an efficient and secure system to achieve privacy-preserving data processing with ABE-based flexible access control. It can support several operations and achieve fine-grained access control without the need of fully trusted cloud servers, which can be deployed in cybersecurity education framework. We also illustrate a number of applications of our system for the purpose of cybersecurity education.

Acknowledgment The work is supported in part by the National Natural Science Foundation of China under Grants 61672410 and 61802293, the National Postdoctoral Program for Innovative Talents under grant BX20180238, the Project funded by China Postdoctoral Science Foundation under grant 2018M633461, the Academy of Finland under Grants 308087, 314203, and 335262, the Shaanxi Innovation Team project under grant 2018TD-007, and the 111 project under grant B16037.

References

1. A. Belle, R. Thiagarajan, S. Soroushmehr, F. Navidi, D.A. Beard, K. Najarian, Big data analytics in healthcare. Biomed. Res. Int. **2015**, 1–16 (2015)
2. G. Javidi, E. Sheybani, K-12 Cybersecurity education, research, and outreach, in *2018 IEEE Frontiers in Education Conference (FIE)*, (San Jose, CA, USA, 2018), pp. 1–5
3. J.J. Stephen, S. Savvides, R. Seidel, P. Eugster, Practical confidentiality preserving big data analysis, in *6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, (Philadelphia, PA, USA, 2014)
4. B. Wang, M. Li, S.S. Chow, H. Li, A tale of two clouds: Computing on data encrypted under multiple keys, in *2014 IEEE Conference on Communications and Network Security (CNS)*, (San Francisco, CA, USA, 2014), pp. 337–345

5. A. Peter, E. Tews, S. Katzenbeisser, Efficiently outsourcing multiparty computation under multiple keys. IEEE Transactions on Information Forensics and Security (TIFS) **8**, 2046–2058 (2013)
6. X. Liu, R. Choo, R. Deng, R. Lu, J. Weng, Efficient and privacy-preserving outsourced calculation of rational numbers. IEEE Transactions on Dependable and Secure Computing (TDSC) **15**, 27–39 (2016)
7. X. Liu, R. Deng, W. Ding, R. Lu, B. Qin, Privacy-preserving outsourced calculation on floating point numbers. IEEE Transactions on Information Forensics and Security **11**, 2513–2527 (2016)
8. R. Bost, R.A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data, in *NDSS*, (San Diego, California, USA, 2015)
9. Z. Yan, P. Zhang, A.V. Vasilakos, A survey on trust management for internet of things. J. Netw. Comput. Appl. **42**, 120–134 (2014)
10. A. Khedr, G. Gulak, SecureMed: Secure medical computation using GPU-accelerated Homomorphic encryption scheme. IEEE Journal of Biomedical & Health Informatics **22**, 597–606 (2017)
11. Z. Brakerski, C. Gentry, V. Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, (Cambridge, MA, USA, 2012), pp. 309–325
12. C. Gentry, Computing arbitrary functions of encrypted data. Commun. ACM **53**, 97–105 (2010)
13. M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, (Riviera, French, 2010), pp. 24–43
14. V.C. Hu, T. Grance, D.F. Ferraiolo, D.R. Kuhn, An access control scheme for big data processing, in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, (Chicago, Illinois, USA, 2014), pp. 1–7
15. Z. Yan, W. Ding, V. Niemi, A.V. Vasilakos, Two schemes of privacy-preserving trust evaluation. Future Generation Computer Systems (FGCS) **62**, 175–189 (2015)
16. C. Huang, Z. Yan, N. Li, M. Wang, Secure pervasive social communications based on Trust in a Distributed way. IEEE Access **4**, 9225–9238 (2016)
17. Z. Yan, X. Li, M. Wang, A. Vasilakos, Flexible data access control based on trust and reputation in cloud computing. IEEE Transactions on Cloud Computing **5**, 485–498 (2015)
18. Z. Yan, X. Li, R. Kantola, Controlling cloud data access based on reputation. Mobile Networks and Applications **20**, 828–839 (2015)
19. W. Ding, Z. Yan, R.H. Deng, Encrypted data processing with Homomorphic re-encryption. Inf. Sci. **409**, 35–55 (2017)
20. J. Feng, L.T. Yang, Q. Zhu, K.-K.R. Choo, Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment. IEEE Transactions on Dependable and Secure Computing (2018). https://doi.org/10.1109/TDSC.2018.2881452
21. L. Kamm, J. Willemson, Secure floating point arithmetic and private satellite collision analysis. Int. J. Inf. Secur. **14**, 531–548 (2015)
22. D. Bogdanov. *Sharemind: Programmable Secure Computations With Practical Applications* (Tartu University, 2013), PhD Thesis
23. J.H. Cheon, J.-S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi, A. Yun, Batch fully homomorphic encryption over the integers, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, (Athens, 2013), pp. 315–335
24. W. Wang, Y. Hu, L. Chen, X. Huang, B. Sunar, Exploring the feasibility of fully homomorphic encryption. IEEE Trans. Comput. **64**, 698–706 (2015)
25. L. Morris, Analysis of partially and fully homomorphic encryption. Rochester Institute of Technology, 1–5 (2013)
26. X. Liu, R.H. Deng, Y. Yang, H.N. Tran, S. Zhong, Hybrid privacy-preserving clinical decision support system in fog–cloud computing. Futur. Gener. Comput. Syst. **78**, 825–837 (2017)

27. Z. Yan, W. Ding, H. Zhu, A scheme to manage encrypted data storage with deduplication in cloud, in *International Conference on Algorithms and Architectures for Parallel Processing*, (Zhangjiajie, China, 2015), pp. 547–561
28. C. Dong, G. Russello, N. Dulay, Shared and searchable encrypted data for untrusted servers, in *IFIP Annual Conference on Data and Applications Security and Privacy*, (London, 2008), pp. 127–143
29. W.C. Garrison III, A. Shull, S. Myers, A.J. Lee, On the practicality of cryptographically enforcing dynamic access control policies in the cloud, in *2016 IEEE Symposium on Security and Privacy*, (San Jose, 2016), pp. 819–838
30. Z. Tianyi, L. Weidong, S. Jiaxing, An efficient role based access control system for cloud computing, in *IEEE 11th International Conference on Computer and Information Technology (CIT)*, (Paphos, Cyprus, 2011), pp. 97–102
31. J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in *2007 IEEE Symposium on Security and Privacy (SP'07)*, (Oakland, 2007), pp. 321–334
32. V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in *13th ACM Conference on Computer and Communications Security*, (Alexandria, 2006), pp. 89–98
33. S. Yu, C. Wang, K. Ren, W. Lou, Achieving secure, scalable, and fine-grained data access control in cloud computing, in *2010 Proceedings IEEE INFOCOM*, (San Diego, 2010), pp. 1–9
34. M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Transactions on Parallel and Distributed Systems **24**, 131–143 (2013)
35. Z. Wan, J.E. Liu, R.H. Deng, HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing. IEEE Transactions on Information Forensics and Security (TIFS) **7**, 743–754 (2012)
36. M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, H. Schröder, Secure computations on non-integer values, in *2010 IEEE International Workshop on Information Forensics and Security*, (Seattle, Washington, USA, 2010), pp. 1–6
37. M. Dahl, C. Ning, T. Toft, On secure two-party integer division, in *International Conference on Financial Cryptography and Data Security*, (Bonaire, 2012), pp. 164–178
38. T. Veugen, Encrypted integer division and secure comparison. International Journal of Applied Cryptography **3**, 166–180 (2014)
39. O. Catrina, A. Saxena, Secure computation with fixed-point numbers, in *International Conference on Financial Cryptography and Data Security*, (Canary Islands, Spain, 2010), pp. 35–50
40. R. Bhoyar, P. Palsodkar, S. Kakde, Design and implementation of goldschmidts algorithm for floating point division and square root, in *International Conference on Communications*, (London, 2015), pp. 1588–1592
41. C. Ugwuoke, Z. Erkin, R.L. Lagendijk, Secure fixed-point division for Homomorphically encrypted operands, in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, (Hamburg, Germany, 2018), pp. 1–10
42. B.K. Samanthula, H. Chun, W. Jiang, An efficient and probabilistic secure bit-decomposition, in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, (Hangzhou, 2013), pp. 541–546
43. R. Gurnani, K. Pandey, S.K. Rai, A scalable model for implementing cyber security exercises, in *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, (New Delhi, 2014), pp. 680–684
44. E. Amankwa, M. Loock, E. Kritzinger, Enhancing information security education and awareness: Proposed characteristics for a model, in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, (Cape Town, 2015), pp. 72–77
45. R. Weiss, F. Turbak, J. Mache, M.E. Locasto, Cybersecurity education and assessment in EDURange. IEEE Security & Privacy **15**(3), 90–95 (2017)

46. J. LeClair, K.M. Hollis, D.M. Pheils, Cybersecurity education and training and its reliance on STEAM, in *2014 IEEE Integrated STEM Education Conference*, (Princeton, NJ, 2014), pp. 1–5
47. M. Frank, M. Leitner, T. Pahi, Design considerations for cyber security Testbeds: A case study on a cyber security Testbed for education, in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, (Orlando, FL, 2017), pp. 38–46
48. A. M'Baya, J. Laval, N. Moalla, Y. Ouzrout, A. Bouras, Ontology based system to guide internship assignment process, in *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, (Naples, 2016), pp. 589–596
49. F. Ghemri, A. Bouras, Innovative education in cyber security field through collaborative education, in *2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, (Bangkok, Thailand, 2018), pp. 1–5
50. R. Verma, M. Kantarcioglu, D. Marchette, E. Leiss, T. Solorio, Security analytics: Essential data analytics knowledge for Cybersecurity professionals and students. IEEE Security & Privacy **13**(6), 60–65 (2015)
51. W.X. Ding, R. Hu, Z. Yan, X.R. Qian, R.H. Deng, L.T. Yang, M.X. Dong, An extended framework of privacy-preserving computation with flexible access control. IEEE Trans. Netw. Serv. Manag., 1 (2019). https://doi.org/10.1109/TNSM.2019.2952462
52. W. Ding, Z. Yan, R. Deng, Privacy-preserving data processing with flexible access control. IEEE Transactions on Dependable & Secure Computing **17**, 363–376 (2017)
53. W.X. Ding, Z. Yan, X.R. Qian, R.H. Deng, Computing maximum and minimum with privacy preservation and flexible access control, in *IEEE GLOBECOM 2019*, (Hawaii, USA, 2019), pp. 1–7
54. P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *International Conference on the Theory and Applications of Cryptographic Techniques*, (Berlin, Germany, 1999), pp. 223–238
55. E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications, in *International Conference on the Theory and Application of Cryptology and Information Security*, (Berlin, Germany, 2003), pp. 37–54