Privacy-Preserving Data Processing with Flexible Access Control

Wenxiu DING, Zheng Yan, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

Abstract—Cloud computing provides an efficient and convenient platform for cloud users to store, process and control their data. Cloud overcomes the bottlenecks of resource-constrained user devices and greatly releases their storage and computing burdens. However, due to the lack of full trust in cloud service providers, the cloud users generally prefer to outsource their sensitive data in an encrypted form, which, however, seriously complicates data processing, analysis, as well as access control. Homomorphic encryption (HE) as a single key system cannot flexibly control data sharing and access after encrypted data processing. How to realize various computations over encrypted data in an efficient way and at the same time flexibly control the access to data processing results has been an important challenging issue. In this paper, we propose a privacy-preserving data processing scheme with flexible access control. With the cooperation of a data service provider (DSP) and a computation party (CP), our scheme, based on Paillier's partial homomorphic encryption (PHE), realizes seven basic operations, i.e., *Addition*, *Subtraction*, *Multiplication*, *Sign Acquisition*, *Absolute*, *Comparison*, and *Equality Test*, over outsourced encrypted data. In addition, our scheme, based on the homomorphism of attribute-based encryption (ABE), is also designed to support flexible access control over processing results of encrypted data. We further prove the security of our scheme and demonstrate its efficiency and advantages through simulations and comparisons with existing work.

Index Terms—Homomorphic encryption, privacy preservation, data sharing, attribute-based encryption

1 Introduction

Cloud computing has been widely adopted in various application domains owing to its specific advantages. It enables cloud users to store their data and perform various computations on the data without incurring a high cost. With the advent of Internet of Things, enormous amounts of data are produced and outsourced to the cloud or cloudlets for storage and analysis. Data analysis helps to gain insights on related entities in a physical world, which can provide tremendous values to various applications in multifarious domains (e.g., medical [1] and business [2]).

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 their processing results, which may seriously impact user privacy. For example, medical case analysis can help in predicting potential illness, but patients may be reluctant to provide their health data due to privacy concerns. Therefore, it is of great significance to protect sensitive data and data processing results from being leaked to any unauthorized parties. A standard solution is to encrypt the data before uploading them to the cloud. However, data encryption introduces several challenges as described below.

First, encryption seriously restricts computations and analyses over the outsourced data in the cloud. With traditional encryption algorithms (e.g., AES), it is impossible for the cloud to process the encrypted data directly. Some existing efforts adopted partial homomorphic encryption (PHE) to solve the problem, but they are limited to multiplication and addition operations on encrypted data [3], [4], which cannot satisfy the demands of many applications. More operations, such as sign acquisition, comparison, absolute and equality test, are expected to be supported in practice [5], [6]. This requests further study on privacy-preserving computations. More basic operations over ciphertexts can obviously support more applications that apply different functions and algorithms, e.g., privacy-preserving classifications in machine learning [7], trust evaluation in Internet of Things [8], and medical analysis in e-health [9]. To realize arbitrary computations over ciphertexts, schemes based on fully homomorphic encryption (FHE) were designed [10], [11], [12]. However, most FHE based schemes suffer from huge computational overhead and high storage cost, which make them impractical for real world deployment and wide use. Currently, the literature still lacks serious studies on efficient computations over ciphertexts.

Second, multi-user access control over ciphertext processing results should also be supported [13]. Existing PHE and FHE schemes are both single-user systems, which inherently lack support on multi-user access to the processing results of encrypted data. The scheme based on PHE [14] supports distribution of addition operation results through an interactive protocol between two servers. But

- *W. Ding is with the State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Changan Qu 710126, China. E-mail: wenxiuding_1989@126.com.*
- *Z. Yan is with the State Key Lab on Integrated Services Networks, School of Cyber Engineering, Xidian University, No.2 South Taibai Road, Xi'an 710071 China, and with the Department of Communications and Networking, Aalto University, Konemiehentie 2, P.O.Box 15400, Espoo 02150, Finland. E-mail: zyan@xidian.edu.cn.*
- *R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 22 July 2017; revised 27 Nov. 2017; accepted 7 Dec. 2017. Date of publication 22 Dec. 2017; date of current version 18 Mar. 2020. (Corresponding author: Zheng Yan.)
Digital Object Identifier no. 10.1109/TDSC.2017.2786247

this protocol must be executed for each data request, thus it is inefficient. Attribute-based encryption (ABE) is an effective tool to support fine-grained access control and multi-user access. It has been applied in many application scenarios [15], [16], [17]. However, to our knowledge, there is no effort in the literature on fine-grained access control over the computation/analysis results based on encrypted data. Our previous work [18] 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 one by one, which obviously incurs high communication and computation costs, as shown in experiments in Section 5.2.3.

In this paper, we propose a novel scheme 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. Specifically, the contributions of this paper can be summarized as follows:

- We propose a generic system architecture consisting of a data service provider (DSP) and a computation party (CP) that seamlessly work together to simultaneously support secure computations over encrypted data and fine-grained access control of computation results.
- We present a family of protocols to efficiently realize seven basic computations over encrypted data: *Addition*, *Subtraction*, *Multiplication*, *Sign Acquisition*, *Absolute*, *Comparison*, and *Equality Test*.
- We propose to utilize ABE with homomorphism to realize fine-grained access control of the processing result of encrypted data, which is not revealed to any system entities including DSP and CP.
- We prove the security of the proposed scheme and demonstrate its efficiency through simulations and comparisons with existing schemes. We show that the proposed scheme is suitable for big data processing. It can be applied in any scenarios with either a small or a large number of data providers.

The rest of this paper is organized as follows. Section 2 gives a brief overview of related work. Section 3 introduces the system model and attack model of our proposed scheme, followed by its detailed design in Section 4. In Section 5, security analysis and performance evaluation are given. Finally, a conclusion is presented in the last section.

2 Related Work

With the development of cloud computing, cloud users benefit from outsourcing data storage and computation to the cloud. However, the risk of personal data disclosure makes it urgent to enhance data security and user privacy. Besides those schemes focusing on data aggregation [19], [20], [21], [22], [23], [24], [25], other studies were conducted to achieve more efficient privacy-preserving operations. In addition, many constructions have been proposed to realize secure access control although the aforementioned issues are still open.

2.1 Secure Data Processing Based on SMC

Secure multi-party computation (SMC) enables computations over multi-user outsourced data without revealing any input. It lays a technical foundation for many problems, such as database query, intrusion detection and data mining with privacy preservation [8]. Several schemes [26], [27] based on the popular SMC construction Sharemind [28] were proposed to achieve various secure computations, e.g., multiplication. But the product of N pieces of data needs about 3^N multiplications of 32-bit numbers under the cooperation of three involved servers, which obviously cannot adapt to big data processing. Although a data requester can easily obtain the final result by requesting all secret preprocessing shares from all involved servers, how to realize fine-grained access control in SMC is still an open issue.

2.2 Secure Data Processing Based on Homomorphic Encryption

FHE schemes [10], [11], [12] are designed to realize arbitrary computations over encrypted data. Due to high computation overhead, some extended schemes [29], [30] were proposed to improve FHE efficiency. However, the computation and storage costs of existing schemes are still not satisfactory for practical use [31], [32].

PHE has been widely used in many applications because it is more efficient and practical than FHE although it can only support limited computations. Some schemes [3], [4] were proposed to support more types of computations, but they can only support addition and multiplication over a limited number of data inputs. In [3], decryption requires solving the problem of discrete logarithm, which seriously restricts the length and the number of data inputs. The multiparty computation framework proposed in [4] achieves addition and multiplication by applying secret sharing. Similar to the SMC-based scheme in [27], it is unable to support the multiplication of a large number of data inputs. Liu et al. [5] proposed a framework for efficient outsourced data calculations with privacy preservation, which can deal with several types of operations, such as addition, multiplication, and division. But this framework cannot support multiplication of a large number of data.

Besides the problems mentioned above, the biggest problem of PHE is that it is a single-user system. This means that the data processing result based on PHE can only be decrypted and accessed by the user with the corresponding secret key of PHE. PHE is not flexible to directly support multi-user access.

2.3 Secure Data Access Control

Cloud storage enables cloud users to upload their data to the cloud for storage and further sharing. However, this causes a new problem that the cloud users lose full control over their data. Thus, an efficient and secure data access control scheme is needed. A number of solutions have been proposed to protect the outsourced data in the cloud, as briefly introduced below.

Proxy re-encryption was adopted to manage data sharing in cloud [33], [34]. But it cannot support fine-grained access control on homomorphic computation results. Role-based access control (RBAC) can only provide partial flexibility based on one level policy, which ensures that only the user with a specified role can access data. But, the constructions [35], [36] based on RBAC cannot support flexible access policies described with various attribute structures. ABE [37],

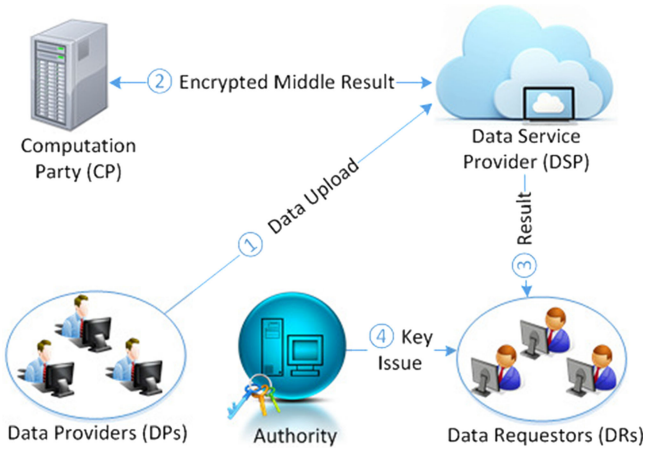

Fig. 1. A system model.

[38] was widely applied in cloud storage management for achieving fine-grained access control [39], [40], [41]. Furthermore, trust-based schemes [15], [16], [17] simplify the attributes involved in ABE and take into consideration only trust levels. These schemes highly reduce computational costs. But, only one system entity (such as one user or one server) is in charge of the access control, which makes this entity obviously knows the contents of results. Thus, it cannot satisfy the demand that the final result cannot be accessed by any unauthorized entity including servers. In this paper, we propose a scheme by taking advantage of the homomorphism of ABE under the same policy to control the access of the processing result of encrypted data based on the cooperation of two servers.

3 Problem Statements

3.1 System Model

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

1) Data service provider (DSP) stores user data, provides some computation service and controls user access. DSP can be operated by a public cloud provider.
2) Computation party (CP) bears the responsibility of computations and access control. It can be served by a private cloud service provider or an administrative department of a company or an institute, which fulfills partial computations and controls access. There may exist multiple CPs for different applications. Each CP provides services for its own consumers. Herein, we simplify our design by considering only one CP in this paper.
3) Data providers (DPs) are the data collectors or producers that encrypt data and store ciphertexts in the DSP for storage and processing.
4) Data requesters (DRs) are the data consumers that acquire the result of data processing/analyzing in a specific context. A DR can also be a DP. DRs are cloud users that take advantage of cloud computing in terms of data storage and data computation. Since the provided data are encrypted, the data processing result is also in an encrypted form. This raises the issue of data access control with regard to the processing results of encrypted data.
5) Authority is fully trusted, which is responsible for system parameter generation and ABE key issuing.

The DPs provide their personal data in an encrypted form and store them at the DSP. Then the DSP cooperates with the CP to complete basic computations over the collected data. In addition, the DSP and CP together execute the access control over the final result of data processing. Only those DRs that satisfy a specific policy can access the final result with the key issued by Authority.

3.2 Attack Model

In the above system, the Authority is regarded as fully trusted to perform its duties, which acts honestly and would never collude with any other entities. All other entities are curious-but-honest. That is, they are curious about others' data, but act honestly by strictly following the design of the system. The DSP and the CP would never collude with each other due to conflict of business interests (e.g., consumer resources and market division) and their legal responsibility. Any their collusion will make them lose reputation, which finally impacts their profits. Herein, we introduce an adversary $\mathcal{A}^*$ in our model, which aims to obtain some specific data by challenging a cloud user (either a DR or a DP) with following capabilities:

1) $\mathcal{A}^*$ may eavesdrop all communication channels to access any encrypted data except the channels between Authority and DRs;
2) $\mathcal{A}^*$ may compromise the DSP (or the CP) to guess the raw data of all ciphertexts outsourced from the DPs, and the raw data of all ciphertexts sent to the CP (or the DSP) and the DR;
3) $\mathcal{A}^*$ may compromise the DSP (or the CP) together with the DPs to guess the final data processing result.
4) $\mathcal{A}^*$ may compromise the DSP (or the CP) together with the DR to guess the raw data from the DP.

The attack model has one restriction: $\mathcal{A}^*$ cannot compromise the challenged DR or DP. The adversary would like to know the raw data of DP (by attacking the DP) or the processing result (by attacking the DR).

4 Privacy-Preserving Data Processing with Access Control

4.1 Notations and Preliminaries

4.1.1 Notations

For easy presentation, Table 1 summarizes the notations used in this paper.

4.1.2 Additive Homomorphic Encryption

Paillier's cryptosystem [42] 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 $[m_i]_{pk}$ $(i = 1, 2, \ldots, N)$. The additive homomorphic encryption satisfies the following equation:

$$D_{sk}\left(\prod_{i=1}^{N} [m_i]_{pk}\right) = \sum_{i=1}^{N} m_i \ ,$$

where $D_{sk}()$ is the corresponding homomorphic decryption algorithm with secret key sk.

TABLE 1
Notation Description

Symbols	Description
g	The system generator that is public;
n	The system parameter;
(sk_{DSP}, pk_{DSP})	The key pair of DSP for data processing;
(sk_{CP}, pk_{CP})	The key pair of CP for data processing;
$PK = pk_{DSP}{}^{sk_{CP}} = pk_{CP}{}^{sk_{DSP}}$	The public parameter based on keys of DSP and CP;
m_i	The raw data provided by DP i;
$[m]$	The ciphertext of m under PK;
$[m]_{pk_i}$	The ciphertext of m under public key pk_i;
$\hat{m}$	The masked message of m;
f	The sign flag in *Sign Acquisition*, *Comparison*, and *Equality Test*;
r	The random value;
N	The number of data providers;
$\mathcal{L}(*)$	The bit length of input data;
$(ck,\ pk_{ck})$	The key pair for final access control;
CK'	The ciphertext of ck using ABE;
ck_1	The partial key for access control chosen by DSP;
ck_2	The partial key for access control chosen by CP;
MSK'	The master secret key in ABE;
SK'	The decryption key in ABE;
PK'	The public key in ABE

4.1.3 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. Generally, KP-ABE [38] consists of four algorithms: *Setup*, *Encrypt*, *KeyGen*, and *Decrypt*.

$Setup^{ABE}(\lambda, U) \rightarrow (PK', MSK')$. The setup algorithm takes in security parameter λ and attribute universe description $U = \{1, 2, \ldots, \omega\}$. It outputs public paramters PK' $(T_1 = g^{t_1}, \ldots, T_{|U|} = g^{t_{|U|}}, Y = e(g', g')^y)$ and master secret key $MSK'(t_1, \ldots, t_{|U|}, y)$, where g' is the generator of G_1 with $e: G_1 \times G_1 \rightarrow G_T$.

$Enc^{ABE}(M, \gamma, PK') \rightarrow CK'$. The encryption algorithm takes in message M, a set of attributes γ and PK'. It outputs ciphertext $CK'(\gamma, E' = MY^s, \{E_i = T_i^s\}_{i\in\gamma})$, where s is a randomly chosen number.

$KeyGen^{ABE}(\mathcal{T}, MSK') \rightarrow SK'$. The key generation algorithm takes in access structure $\mathcal{T}$, the master secret key MSK'. It outputs decryption key SK' ($SK'_i = g'^{q_x(0)/t_i}$ where $i = att(x)$, $q_x(0) = q_{parent(x)}(index(x))$) and $q_r(0) = y$.

$Dec^{ABE}(CK',\ PK',\ SK') \rightarrow M$. The decryption algorithm takes in PK', SK' and the ciphertext CK'. If the set of attributes satisfies the access policy tree $\mathcal{T}$ embedded in the private key, it finally outputs the message M.

For more details about KP-ABE, refer to [38]. Notably, ciphertext-policy attribute-based encryption (CP-ABE) [37] can also be applied to implement our scheme. The KP-ABE is multiplicative homomorphic if the same attributes are employed to encrypt two pieces of raw data. That is, given two ABE ciphertexts of M_1 and M_2 under the same policy, the ciphertext of $M_1 * M_2$ can be obtained through the multiplication of two ciphertexts $Enc^{ABE}(M_1, \gamma, PK') * Enc^{ABE}(M_2, \gamma, PK')$, marked as HE^{ABE}. The length of raw data is limited and highly related to the system parameters. In this paper, we employ this feature to realize the secure access control that can prevent the reveal of processing result to any involved entities. Refer to Section 5.2.3 for more details.

4.2 Homomorphic Re-Encryption Scheme (HRES)

In order to support privacy-preserving data processing, we revise the scheme [43] (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 our previous work [18].

*Key Generation (**KeyGen**).* Let k be a security parameter and p, q be two large primes, where $\mathcal{L}(p) = \mathcal{L}(q) = k$ ($\mathcal{L}(\cdot)$ returns the bit length of input data). Due to the property of safe primes, there exist two primes p' and q' which satisfy that $p = 2p' + 1$, $q = 2q' + 1$. We compute $n = p*q$ and choose a generator g with order $\lambda = 2p'q'$, which can be chosen by selecting a random number $z \in \mathbb{Z}^*_{n^2}$ and computing $g = -z^{2n}$. The value λ can be used to decrypt the encrypted data, but we choose to conceal it and protect it from all involved parties. In the HRES, we only use key pair (sk, g^{sk}) for data encryption and decryption. The DSP and the CP generate their key pairs:$(sk_{DSP} = a, pk_{DSP} = g^a)$ and $(sk_{CP} = b, pk_{CP} = g^b)$, and then negotiate their Diffie-Hellman key $PK = pk_{DSP}{}^{sk_{CP}} = pk_{CP}{}^{sk_{DSP}} = g^{a*b}$. To support encrypted data processing, PK is public to all involved parties. At system setup, cloud user i (i.e., DP or DR) generates its key pair $(sk_i, pk_i) = (k_i, g^{k_i})$. The public system parameters include $\{g, n, PK\}$.

First, the original encryption is directly obtained from [43], which is a general public key cryptosystem.

Encryption **(*Enc*)**. Any entity (e.g., cloud user or cloud server) wants to send its data to a specific cloud user i. It simply encrypts its data with the pubic key of cloud user i (pk_i) and random $r \in [1, n/4]$, and sends ciphertext to cloud user i:

$$[m]_{pk_i} = \{(1 + m * n)pk_i{}^r,\ \ g^r\}\ \big(mod\ n^2\big).$$

Decryption **(*Dec*)**. Upon receiving the encrypted data, cloud user i can directly decrypt it to obtain the original data:

$$m = L\Big((1 + m * n)pk_i{}^r/(g^r)^{k_i} mod\ n^2\Big).$$

Second, a *Two-Level Decryption* scheme is newly designed to flexibly support outsourced data processing, as presented below:

Encryption with Two Keys **(*EncTK*)**. Given message $m_i \in \mathbb{Z}_n$ provided by cloud user i, we first select random number $r \in [1, n/4]$ and then encrypt it with PK generated from the keys of two servers. The ciphertext is generated as $[m_i] = [m_i]_{PK} = \{T_i, T_i'\}$, where $T_i = (1 + m_i * n) * PK^r\ mod\ n^2$ and $T_i' = g^r\ mod\ n^2$.

Note: for ease of presentation, we use $[m_i]$ to denote the ciphertext of m_i encrypted with PK, which can only be decrypted under the cooperation of DSP and CP.

Partial Decryption with SK_{DSP} **(*PDec*1)**. Once $[m_i]$ is received by the DSP, algorithm ***PDec***1 will be run to transfer it into another ciphertext that can be decrypted by the CP as follows:

$$\begin{aligned}[m_i]_{pk_{CP}} &= \left\{T_i^{(1)}, T_i'^{(1)}\right\} = \left\{T_i, (T_i')^{sk_{DSP}}\right\} \\ &= \{(1 + m_i * n)PK^r, g^{r*a}\}\ mod\ n^2 \\ &= \{(1 + m_i * n)pk_{CP}{}^{a*r}, g^{r*a}\}\ mod\ n^2.\end{aligned}$$

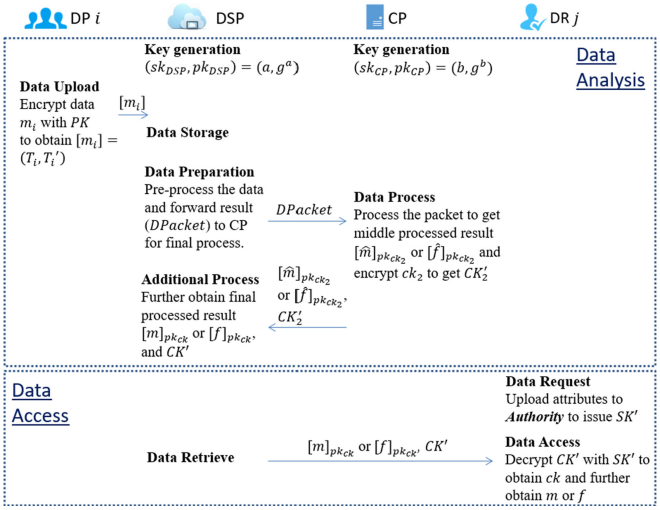

Fig. 2. A brief procedure of data processing.

Partial Decryption with SK_{CP} (PDec2). Once the encrypted data $[m_i]_{pk_{CP}}$ is received, the CP can directly decrypt it with its own secret key as follows:

1) $T_i'^{(2)} = (T_i'^{(1)})^{sk_{CP}} = g^{r*a*b} = PK^r \ mod \ n^2$;
2) $m_i = L(T_i^{(1)} / T_i'^{(2)} \ mod \ n^2)$, where $L(u) = (u-1)/n$.

In addition, the encrypted data under any public key pk has the following properties:

1) Additive homomorphism:
$[m_1]_{pk} * [m_2]_{pk} = [m_1 + m_2]_{pk}$;

2) $([m]_{pk})^t = \{\{(1+m*n)pk^r\}^t, (g^r)^t\} \ mod \ n^2$
$= \{(1+m*n)^t pk^{r*t}, g^{r*t}\} \ mod \ n^2$
$= \{(1+t*m*n)pk^{r*t}, g^{r*t}\} \ mod \ n^2$
$= [t*m]_{pk}$;

3) $([m]_{pk})^{n-1} = \{\{(1+m*n)pk^r\}^{(n-1)}, (g^r)^{(n-1)}\} \ mod \ n^2$
$= \{(1+m*(n-1)*n)pk^{r(n-1)}, g^{r(n-1)}\} \ mod \ n^2$
$= \{(1-m*n+m*n^2)pk^{r(n-1)}, g^{r(n-1)}\} \ mod \ n^2$
$= \{(1-m*n)pk^{r(n-1)}, g^{r(n-1)}\} \ mod \ n^2$
$= [-m]_{pk}$.

4.3 Data Processing Procedure

The brief procedure of data processing is shown in Fig. 2. It mainly involves six steps:

Step 1 (System Setup @ All Entities). Authority calls the algorithm $KeyGen$ in Section 4.2 and $Setup^{ABE}(\lambda, U)$ in Section 4.1.3 to complete the setup of HRES and ABE. Note: if multiple CPs are employed in the system, each CP can negotiate a Diffie-Hellman key PK with the DSP and publish this key to its customers.

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, $|m_i| < \mathcal{L}(n)/4$):

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

Step 3 (Data Preparation @ DSP). Upon receiving the data from DPs, the DSP needs to do some analyses over the encrypted data. It pre-processes the data and provides a data packet $DPacket$ and ABE ciphertext for access control to the CP. The details of this process differ in various operations and are given in the next sub-section. 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 pre-processing results from the DSP, the CP chooses another random partial key ck_2 and further processes data to obtain the pre-processing result $[\hat{m}]_{pk_{ck_2}}$ or $[\hat{f}]_{pk_{ck_2}}$.

Regarding to access control, CP encrypts ck_2 using ABE to get $CK_2' = Enc^{ABE}(ck_2, \gamma, PK')$ and forwards it to DSP.

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

Regarding to access control, the DSP encrypts ck_1 using ABE under the same policy to get CK_1' and further gets CK' through the homomorphism of ABE:

$$CK' = CK_1' * CK_2' = Enc^{ABE}(ck_1 * ck_2, \gamma, PK').$$

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 to meet their computation demand.

4.4 Detailed Data Processing

System setup and data collection are the same as those in Section 4.3. The operations for access control are also the same as those above, thus we do not introduce the details in this section.

Our proposed scheme supports seven basic operations over encrypted data: 1) *Addition*; 2) *Subtraction*; 3) *Multiplication*; 4) *Sign Acquisition*; 5) *Absolute; 6) Comparison*; and 7) *Equality Test*. In the following sub-sections, we mainly focus on the steps from 3 to 5 in each basic operation.

4.4.1 Addition

Addition obtains 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:

$$[m] = (T, T') = \prod_{i=1}^{N} [m_i] \ = \left(\prod_{i=1}^{N} T_i, \prod_{i=1}^{N} T_i'\right)$$

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:

$$[c_1(m+r_1)] = \left(\tilde{T}, \tilde{T'}\right) = \{(T(1+r_1*n))^{c_1}, \ (T')^{c_1}\}$$

3) Call ***PDec***1 to partially decrypt it:

$$[c_1(m+r_1)]_{pk_{CP}} = \left(\hat{T}, \hat{T'}\right) = \left\{\tilde{T}, \ \left(\tilde{T'}\right)^a\right\}$$

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 finally decrypt the encrypted data and obtain $c_1(m+r_1)$. And then the CP chooses the second partial key ck_2 and a random number r to encrypt data as follows:

$$[c_1(m+r_1)]_{pk_{ck_2}} = \left(\bar{T}, \overline{T'}\right) = \left\{(1+c_1(m+r_1)n)g^{ck_2 * r}, g^r\right\}$$

where $pk_{ck_2} = g^{ck_2}$.

Then the CP encrypts ck_2 to obtain CK_2' and forwards $[\hat{m}]_{pk_{ck}}$ and CK_2' back to the DSP.

Step 5 (Additional Process @ DSP). The DSP computes to obtain the final encrypted processing result with ck_1 and r_1:

$$[m]_{pk_{ck}} = \left(\bar{T}^{ck_1}(1-r_1 n), \overline{T'}\right) = \left\{(1+m*n)g^{ck_1 * ck_2 * r},\ g^r\right\}$$

where $pk_{ck} = g^{ck_1 * ck_2}$ and $ck = ck_1 * ck_2$.

Similar to Section 4.3, it encrypts ck_1 using ABE and gets $CK' = CK_1' * CK_2' = Enc^{ABE}(ck_1 * ck_2, \gamma, PK')$.

4.4.2 Subtraction

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

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

4.4.3 Multiplication

Multiplication obtains the product of all raw data $(m = \prod_{i=1}^{N} m_i)$. Multiplication is a bit more complicated than *Addition*. It can be accomplished with three steps: 1) mask the raw data by raising to the power of a random number c; 2) decrypt all masked ciphertexts, multiply them in the form of plaintexts and then re-encrypt masked result; 3) encrypt masked multiplication result and then remove the mask. For ease of presentation, we describe the details with two pieces of data ($[m_1]$, $[m_2]$). The DR wants to get the multiplication result $m = m_1 * m_2$.

Note that the available number of the data in multiplication influences the length of original raw data. If we need to get the product of N pieces of data, it must be guaranteed that $\mathcal{L}(m_i) < \mathcal{L}(n)/(2N)$, which is different from *Addition*.

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 ***PDec***1:

1) $[c_1 * m_1] = \{T_1{}^{c_1}, (T_1')^{c_1}\}$;
2) $[c_1 * m_1]_{pk_{CP}} = (T_1{}^{(1)}, T_1{}'^{(1)}) = \{T_1{}^{c_1}, (T_1')^{c_1 * a}\} = \{(1 + c_1 * m_1 * n) * PK^{r_1 * c_1},\ g^{r_1 * a * c_1}\}$;
3) $[c_2 * m_2] = \{T_2{}^{c_2}, (T_2')^{c_2}\}$;
4) $[c_2 * m_2]_{pk_{CP}} = (T_2{}^{(1)}, T_2{}'^{(1)}) = \{T_2{}^{c_2}, (T_2')^{c_2 * a}\} = \{(1 + c_2 * m_2 * n) * PK^{r_2 * c_2},\ g^{r_2 * a * c_2}\}$.

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 ***PDec***2 to decrypt the data:

$$c_1 * m_1 = T_1{}^{(1)} / \left(T_1{}'^{(1)}\right)^b, c_2 * m_2 = T_2{}^{(1)} / \left(T_2{}'^{(1)}\right)^b.$$

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:

1) $[\hat{m}]_{pk_{ck_2}} = [c_1 c_2 m]_{pk_{ck_2}} = (\bar{T}, \overline{T'}) = \{(1 + c_1 m_1 c_2 m_2 * n) g^{ck_2 * r}, g^r\}$;
2) $CK_2' = Enc^{ABE}(ck_2, \gamma, PK')$.

Finally, the CP forwards $[\hat{m}]_{pk_{ck_2}}$ and CK_2' to the DSP.

Step 5 (Additional Process @ DSP). The DSP further processes the data packet with ck_1 and then gets ciphertext of key as follows:

1) $[m]_{pk_{ck}} = \{\bar{T}^{ck_1}, \overline{T'}\} = \{(1 + m * n) g^{ck_1 * ck_2 * r},\ g^r\}$;
2) $CK' = CK_2' * Enc^{ABE}(ck_1, \gamma, PK')$.

4.4.4 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 $[-BIG, BIG]$. DR wants to know the sign of raw data m_1 from $[m_1]$. *Sign Acquisition* can be achieved by masking the original ciphertexts 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 $(\mathcal{L}(R) < \mathcal{L}(n)/4)$, c_1 and ck_1. It first encrypts "1" and then computes as follows:

1) $[1] = \{(1+n) * PK^{r'}, g^{r'}\}$;
2) $[2 * m_1 + 1] = (T, T') = [m_1]^2 * [1] = \{(1 + (2 * m_1 + 1) * n) * PK^{r' + 2 * r_1},\ g^{r' + 2 * r_1}\}$;
3) Then it flips a coin s. If $s = -1$; it computes:

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

4) Otherwise $(s = 1)$, it calls ***PDec***1 and computes: $[m']_{pk_{CP}} = \{T^R, T'^{a*R}\} = [R * (2 * m_1 + 1)]$.
5) The DSP computes $c_2 = (ck_1)^{-1} mod\ n$, and $s' = c_1 * c_2 * s\ mod\ n$.

The data packet sent to the CP is $\{[m']_{pk_{CP}},\ s'\}$.

Step 4 (Data Process @ CP). Upon receiving the data packet from the DSP, the CP decrypts $[m']_{pk_{CP}}$ with ***PDec***2 to obtain raw data m'.

The CP compares $\mathcal{L}(m')$ with $\mathcal{L}(n)/2$. If $\mathcal{L}(m') < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$.

The CP chooses a random number r and a second partial key ck_2, and further computes as follows:

1) $[\hat{f}]_{pk_{ck_2}} = (\bar{T}, \overline{T'}) = \{(1 + s' u * n) g^{ck_2 * r}, g^r\}$;
2) Encrypt ck_2 using ABE: $CK_2' = Enc^{ABE}(ck_2, \gamma, PK')$.

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

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

1) Compute $c_3 = {c_1}^{-1} mod\ n$;
2) $[f]_{pk_{ck}} = \{\bar{T}^{ck_1 * c_3}, (\overline{T'})^{c_3}\} = \{(1 + su * n) g^{ck_1 * ck_2 * r * c_3},\ g^{r * c_3}\}$.
3) $CK' = Enc^{ABE}(ck_1, \gamma, PK') * CK'_2$.

Step 6 (Data Access @ DR). The DR satisfying the access policy in ABE can decrypt CK' to obtain ck and further decrypt $[f]_{pk_{ck}}$ to obtain f.

Note: if $f = 1$, $m_1 \geq 0$; Otherwise, $m_1 < 0$.

4.4.5 Absolute

Besides the operations in *Sign Acquisition*, *Absolute* needs to mask the original data by raising to the power of a random number, and then remove the mask according to sign indicator to achieve a final absolute result. 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 also chooses the first partial key ck_1. It first encrypts "1" and then computes as follows:

1) $[1] = \{(1 + n) * PK^{r'}, g^{r'}\}$;
2) $[2 * m_1 + 1] = (T, T') = [m_1]^2 * [1] = \{(1 + (2 * m_1 + 1) * n) * PK^{r' + 2 * r_1},\ g^{r' + 2 * r_1}\}$;
3) Then it flips a coin s. If $s = -1$; it computes:

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

Otherwise ($s = 1$), it calls ***PDec***1 and computes: $[m']_{pk_{CP}} = \{T^R, T'^{a*R}\} = [R * (2 * m_1 + 1)]$;

4) Compute $[c_1 m_1] = [m_1]^{c_1}$, and call ***PDec***1 to obtain $[c_1 m_1]_{pk_{CP}}$.
5) The DSP sets $c_3 = (ck_1)^{-1} mod\ n$, and $s' = c_2 * c_3 * s\ mod\ n$.

The data packet sent to the CP is $\{[m']_{pk_{CP}},\ s',\ [c_1 m_1]_{pk_{CP}}\}$.

Step 4 (Data Process @ CP). Upon receiving the data packet from the DSP, the CP decrypts $[c_1 m_1]_{pk_{CP}}$ and $[m']_{pk_{CP}}$ with ***PDec***2 to obtain $c_1 m_1$ and m', respectively. The 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:

1) $[c_1 m_1 s' u]_{pk_{ck_2}} = (\bar{T}, \overline{T'}) = \{(1 + c_1 m_1 s' u * n) g^{ck_2 * r}, g^r\}$;
2) Encrypt ck_2 using ABE: $CK'_2 = Enc^{ABE}(ck_2, \gamma, PK')$.

Finally, the CP forwards $[c_1 m_1 s' u]_{pk_{ck_2}}$ 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 $c_5 = (c_2)^{-1} mod\ n$;
2) $[su * m_1]_{pk_{ck}} = \{\bar{T}^{ck_1 * c_4 * c_5}, \overline{T'}^{c_4 * c_5}\} = \{(1 + su * m_1 * n)\ g^{ck_1 * ck_2 * r * c_4 * c_5},\ g^{r * c_4 * c_5}\}$;
3) $CK' = Enc^{ABE}(ck_1, \gamma, PK') * CK'_2$.

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.

4.4.6 Comparison

Comparison can be simply accomplished by checking the sign of the difference value of two data by calling *Sign Acquisition*. Similar to the functions above, DR wants to compare the raw data (m_1, m_2) based on their encrypted data. For ease of presentation, $m_1 - m_2$ is denoted as m_{1-2}.

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

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

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

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

4.4.7 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 the encrypted data ($[m_1]$, $[m_2]$). The DSP and the CP directly interact with each other in two parallel computations of *Comparison*.

They compare m_1 and m_2 in two forms: 1) $m_{1-2} = m_1 - m_2$; 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, the DR that satisfies the access policy in ABE can decrypt CK' to obtain ck. The 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, $m_1 \neq m_2$.

4.5 Data Analysis over Fixed Point Numbers

In order to adapt to various applications, we further design a scheme to deal with fixed point numbers rather than integers.

Normally, a fixed-point number can be represented with three parts: the sign field, integer field and fractional field. Given a fixed point number m, we set it in binary format as $(S, b_{e-2}, b_{e-3}, \ldots, b_0, b_{-1}, \ldots, b_{-f})$, where S is its sign, $e - 1$ is the length of the integer field, f is the length of the fractional field and its value is $m = S * \sum_{i=-f}^{e-2} (b_i * 2^i)$. In order to encrypt number m, we should first scale $|m|$ to $|m'| = |m| * 2^f$. If m is a positive value, then we can set $m' = m * 2^f$; otherwise, we should set $m' = N - m * 2^f$. Then m' as an integer can be encrypted as $[m']$.

Regarding to the operations presented in Section 4.4 except multiplication, it is easy to be executed over fixed point numbers.

4.5.1 Multiplication over Encrypted Fixed Point Numbers

For two fixed point numbers m_1 and m_2, we perform as follows:

1) Get the integer representation of m_1 and m_2: $m'_1 = m_1 * 2^f$ and $m'_2 = m_2 * 2^f$;
2) Users upload their ciphertext to CSP as: $[m'_1]$ and $[m'_2]$.
3) Then execute Multiplication over $[m'_1]$ and $[m'_2]$;
4) Finally, DR can get the value of $\mathrm{m}'_1 * \mathrm{m}'_2$, then scale down to obtain the final result $m_1 * m_2 = m'_1 * m'_2 * 2^{-f}$.

4.5.2 Polynomial Computations over Encrypted Fixed Point Numbers

For a clear presentation, we give an example of $m_1 * m_2 + m_3$. The detailed procedure is presented as follows:

1) (@DPs) Get the integer representation of three numbers: $m'_1 = m_1 * 2^f$, $m'_2 = m_2 * 2^f$ and $m'_3 = m_3 * 2^f$;
2) (@DSP) Execute the same operations to the Step 3 in Multiplication to get: $\{[c_1 * m'_1]_{pk_{CP}}, [c_2 * m'_2]_{pk_{CP}}\}$.
3) (@CP) Decrypt data packet to get $c_1 * m'_1$ and $c_2 * m'_2$; Then encrypt $c_1 * m'_1 * c_2 * m'_2$ with Diffie-Hellman key PK to get $[c_1 c_2 * m'_1 m'_2]$; Send it to DSP.
4) (@DSP) First compute $[m'_3]^{2^f}$ to scale the original data and get $[2^f * m'_3]$; Execute the proposed scheme *Addition* over $[2^f * m'_3]$ and $[c_1 c_2 * m'_1 m'_2]$ under the cooperation with CP.

Note: The length of data encrypted via Paillier system is less than $\mathcal{L}(n)$. In order to support various computations above over positive and negative numbers, we strictly restrict the length of data such that $\mathcal{L}(m) < 1/2 * \mathcal{L}(n)$. If multiplication of N pieces of data is needed, then it has a strict limitation that $\mathcal{L}(m) = (e + f) < \mathcal{L}(n)/(2N)$. Here, with the default setting $\mathcal{L}(n) = 1024$, $e = 45$, $f = 15$, we should limit the multiplication times (i.e., the number of provided data) to 8.

5 Security Analysis and Performance Evaluation

5.1 Security Analysis

The semantic security of HRES has been proved in our previous work [18]. Hence, we skip its security proof and focus on the security analysis of our proposed schemes. Here, we adopt the security model for securely realizing an ideal functionality in the presence of semi-honest (non-colluding) adversaries. It involves four types of parties: DSP, CP, DP and DR. Thus, we construct four simulators $Sim = (Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$ to against four kinds of adversaries $(\mathcal{A}_{DP}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DR})$ that corrupt DP, DSP, CP and DR, respectively.

Theorem 1. *The* Addition *scheme in Section 4.4.1 can securely obtain the plaintext of addition via computations on ciphertexts in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{DR}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DP})$.

Proof. We present the construction of four independent simulators $(Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$. Here, we prove the security of the case with two inputs (i.e., $N = 2$). □

Sim_{DP} receives the input of m_1 and m_2, then it simulates $\mathcal{A}_{DP}$ as follows: it encrypts data m_1 as $[m_1] = EncTK(m_1)$, and data m_2 as $[m_2] = EncTK(m_2)$. Finally, it returns $[m_1]$ and $[m_2]$ to $\mathcal{A}_{DP}$, and outputs the entire view of $\mathcal{A}_{DP}$.

The view of $\mathcal{A}_{DP}$ is the encrypted data. The views of $\mathcal{A}_{DP}$ in the real and the ideal executions are indistinguishable.

Sim_{DSP} simulates $\mathcal{A}_{DSP}$ as follows: it runs the ***EncTK*** on two randomly chosen numbers $\widetilde{m_1}$ and $\widetilde{m_2}$; then it multiplies $[\widetilde{m_1}]$ by $[\widetilde{m_2}]$ to get $[\tilde{m}]$ where $m = \widetilde{m_1} + \widetilde{m_2}$; further it masks the ciphertext with two random numbers r_1 and ck_1, and runs the ***PDec***1 to obtain $[c_1(\tilde{m} + r_1)]_{pk_{CP}}$. By accessing Sim_{CP}, it receives $[\widehat{\widetilde{m}}]_{pk_{ck_2}}$ and $\widehat{CK'_2}\$$ based on a randomly generated key ck_2. Then it computes to obtain $[\tilde{m}]_{pk_{ck}}$ and $\widetilde{CK'}$. Finally, Sim_{DSP} outputs $[c_1(\tilde{m} + r_1)]_{pk_{CP}}$, $[\widehat{\widetilde{m}}]_{pk_{ck_2}}$, $\widehat{CK'_2}$, $[\tilde{m}]_{pk_{ck}}$ and $\widehat{CK'}$ to $\mathcal{A}_{DSP}$. If $\mathcal{A}_{DSP}$ replies with $\perp$, Sim_{DSP} returns $\perp$.

The view of $\mathcal{A}_{DSP}$ consists of the encrypted data and the ciphertext of corresponding decryption key. Owing to the honesty of the challenged cloud user and the security of the HRES, $\mathcal{A}_{DSP}$ receives the same output in both real and ideal executions. As the decryption key ck is randomly constructed by CSP and CP in each challenge, $\mathcal{A}_{DSP}$ would not obtain more information by executing multiple challenges. Thus, the views are indistinguishable.

Sim_{CP} simulates $\mathcal{A}_{CP}$ as follows: it randomly chooses data $\hat{m}$, uses ***Enc*** to obtain $[\hat{m}]_{pk_{ck_2}}$ and calls ABE encryption to obtain $\widehat{CK'_2}$. Then it sends $[\hat{m}]_{pk_{ck_2}}$ and $\widehat{CK'_2}$ to $\mathcal{A}_{CP}$. If $\mathcal{A}_{CP}$ replies with $\perp$, Sim_{CP} returns $\perp$. In both real and ideal executions, it receives the output of two ciphertexts. In the real world, the security is guaranteed by the semantic security of HRES and the security of ABE.

Sim_{DR} simulates $\mathcal{A}_{DR}$ as follows: (it cannot enquire for the challenged data) it randomly chooses data $[m']_{pk_{ck}}$ and decrypts it to obtain m', and then sends it to $\mathcal{A}_{DP}$. If $\mathcal{A}_{DP}$ replies with $\perp$, Sim_{DR} returns $\perp$.

The view of $\mathcal{A}_{DR}$ is the decrypted result without any other information. But in both the real and ideal executions, it is guaranteed by the semantic security of the HRES. The views are indistinguishable in both executions.

No matter how many times the adversary accesses the simulator $\mathcal{A}_{DR}$, it is still difficult to obtain the original real data for two reasons: 1) the randomly chosen data have no relation to the original real data; 2) exhaustion attack is hard owing to the randomness of the chosen numbers.

The security proofs of other operations are similar to that of the *Addition* under the semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{DR}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DP})$.

5.2 Performance Evaluation

In this section, we analyze the computational complexity and the communication overhead of our proposed seven computing operation schemes. Further, we implemented them and tested their performances through simulations.

5.2.1 Computational Complexity Analysis

Herein, we analyze the computational complexity of our designed schemes. Notably, the ABE and the HRES are based on different cryptographic techniques, while the computational complexity by employing ABE to control access in each operation is the same. Hence, we analyze the computational complexity of ABE in a separate part.

1) The Computational Complexity of ABE. As presented in Section 4.1.3, the computations of all four algorithms are related to the number of attributes. In order to clarify the relationships, we assume that there are $|U|$ universal attributes, that $|\gamma|$ attributes are in the access policy tree $\mathcal{T}$, and that at most ϑ attributes should be satisfied in the policy tree $\mathcal{T}$ to decrypt ciphertext.

The setup algorithm $Setup^{ABE}()$ should choose the system parameters and generate the public parameters. It needs to do $|U|+1$ exponentiations and one bilinear pairing. The encryption algorithm $Enc^{ABE}()$ needs to generate the ciphertext by involving each attribute in $\mathcal{T}$ with one exponentiation, and encrypt the original message with one exponentiation. Hence, the encryption algorithm needs $|\gamma|+1$ exponentiations in total. $KeyGen^{ABE}()$ operates $|\gamma|+1$ exponentiations to generate decryption key. $HE^{ABE}()$ merely involves $|\gamma|+1$ multiplications. As exponentiation is significantly more costly than multiplication, we ignore the computation cost of $HE^{ABE}()$ in our anlysis. The main operation in $Dec^{ABE}()$ is to compute the divided pieces of encrypted partial keys, which performs at most ϑ bilinear pairings.

The system setup and data encryption only need to be executed once. Thus, their computational complexity can be amortized when multiple users access the final result. Though the computation cost of cloud users in decryption algorithm is higher than our previous scheme [18], it enhances the security of processing results by introducing fine-grained access control. The system should balance efficiency and security according to its capabilities and requirements. Moreover, trust can be used as a single attribute in ABE in order to greatly reduce computational cost [15], [16], [17].

2) The Computational Complexity of Data Analysis. As the computations related to ABE for access control neither vary with the number of provided data nor vary with the designed functions, their costs are ignored in the following analysis.

The modular exponentiation operation is significantly more time-consuming than the modular addition and multiplication, thus we ignore the fixed numbers of additions and multiplications in the following analysis. In addition, the computational complexity of basic operations (modular exponentiation in HRES, exponentiation and bilinear pairing in ABE) would never be affected by the number of provided data or access policy. Thus, we set all their computational complexities to be $\mathcal{O}(1)$.

Enc and ***EncTK*** need two modular exponentiations. Both the algorithms ***Dec*** and ***PDec***2 take one modular exponentiation and one modular multiplication to obtain the ciphertext, and ***PDec***1 needs one modular exponentiation. Based on these analyzed results, we further give the computational complexity in seven operation functions. We hold the assumption that there are N pieces of provided data in *Addition*, *Subtraction*, and *Multiplication*.

Computational Complexity of DP. The DP directly outsources its data with ***EncTK*** for data processing/analyzing. Its computation cost of outsourcing one piece of data is two modular exponentiations, which is with the computational complexity $\mathcal{O}(1)$.

Computational Complexity of DSP. In *Addition*, the DSP has two parts of computations in each algorithm. In Step 3, the DSP first needs to multiply the N pieces of data, mask ciphertexts with random numbers through two modular exponentiations, and then call $PDec1$. Moreover, it should do one modular exponentiation for removing the mask in Step 5. In *Subtraction*, the DSP needs to obtain the negative of subtractor with two more modular exponentiations than that in *Addition*. As the modular exponentiation of g^r with $r \in [1,n]$ needs at most n modular multiplications, the multiplication of N ($N < n$) pieces of provided data results in computational complexity of $\mathcal{O}(1)$. Thus, the DSP has the computational complexity of $\mathcal{O}(1)$ in both *Addition* and *Subtraction*.

In *Multiplication*, the DSP should first mask each piece of data with a random number through two exponentiations and then call ***PDec***1 to partially decrypt the data in Step 3, which results in $3N$ modular exponentiations. In Step 5, it should do one exponentiation to remove the mask from original data. Hence, the computational complexity of DSP results in $\mathcal{O}(N)$.

In *Sign Acquisition*, the DSP needs to do four modular exponentiations in Step 3, and two more in Step 5. In *Absolute*, the DSP needs to do three more modular exponentiations in Step 3 than it does in *Sign Acquisition*. In *Comparison*, the DSP should first get the subtraction of two pieces of encrypted data, which needs two modular exponentiations, and then do the same operation as it does in *Sign Acquisition*. In *Equality Test*, the DSP needs invoke the *Comparison* twice. Hence, its computational complexity in these three schemes are all $\mathcal{O}(1)$.

Computational Complexity of CP. In the schemes except *Multiplication*, *Absolute* and *Equality Test*, the CP first decrypts ciphertext by calling ***PDec***2 and then encrypts the data with a partial secret key, which needs three modular exponentiations in total. Hence, the computational complexities in them are all $\mathcal{O}(1)$ regardless of the ABE encryption.

In *Multiplication*, the CP should first decrypt each ciphertext of masked original data with ***PDec***2, then encrypt their product with ***Enc***. It totally needs $(N+2)$ modular exponentiation and results in computational complexity $\mathcal{O}(N+2)$.

Different from the analysis above, the CP in *Absolute* should first do two partial decryptions $PDec2$. Generally, it needs four modular exponentiations. Its computational complexity results in $\mathcal{O}(1)$.

In *Equality Test*, the CP takes the double computation costs in *Comparison*, which needs six modular exponentiations but still results in the computational complexity of $\mathcal{O}(1)$.

Computational Complexity of DR. The DR should first call $Dec^{ABE}()$ to obtain the decryption key ck. Then, it further decrypts the pre-processing result to obtain the final result by calling ***Dec***, which needs one modular exponentiation.

3) The Summary of Computation Analysis in Seven Operation Schemes. In the following tables and figures, we use the following corresponding notations: *Addition* (Add), *Subtraction* (Sub.), *Multiplication* (Mul.), *Sign Acquisition* (Sign), *Absolute* (Abs.), *Comparison* (Comp.), and *Equality Test* (Equal). Here, we present the total computation costs in each algorithm and compare them with existing work [18] in Table 2. The constant number of multiplications is not considered in the table. We can observe that our new schemes introduce some computation cost to the involved entities, but it can greatly reduce the cost of DSP and CP when the number of DRs is high, especially in *Multiplication*, which is further discussed in Section 5.2.2. Moreover, our scheme provides fine-grained access control and enhances the security of processing results.

TABLE 2
Computation Analysis

Role	Operation	Computations of our work	Complexity of our work	Computations in [18]
DP	ALL	$2 * ModExp$	$\mathcal{O}(1)$	$2 * ModExp$
DSP	Add	$N * ModMul + 4 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$(N * ModMul + 3ModExp)N_R$
	Sub.	$N * ModMul + 6 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$(N * ModMul + 5ModExp)N_R$
	Mul.	$(3N + 1)ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(N + \|\gamma\|)$	$N_R(2N + 2) * ModExp$
	Sign	$6 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$3N_R * ModExp$
	Abs.	$9 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	–
	Comp.	$8 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$4N_R * ModExp$
	Equal	$16 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$8N_R * ModExp$
	Vari.	–	–	$N_R(4N + 2) * ModExp$
CP	Add	$3 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$3N_R * ModExp$
	Sub.	$3 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$3N_R * ModExp$
	Mul.	$(N + 2)ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(N + \|\gamma\|)$	$N_R(N + 2) * ModExp$
	Sign	$3 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$3N_R * ModExp$
	Abs.	$4 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	–
	Comp.	$3 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$3N_R * ModExp$
	Equal	$6 * ModExp + (\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	$6N_R * ModExp$
	Vari.	–	–	$N_R((N + 2) * ModExp + (\|\gamma\| + 1)Exp))$
Authority	ALL	$N_R(\|\gamma\| + 1)Exp$	$\mathcal{O}(\|\gamma\|)$	–
DR	ALL	$1 * ModExp + \vartheta * BiPair$	$\mathcal{O}(\vartheta)$	$1 * ModExp$

Notes: N: the number of provided data; $BiPair$: the bilinear pairing in ABE; Exp : the exponentiation in ABE; $ModExp$: the modular exponentiation in HRES; $ModMul$: the modular multiplication; ALL: fits all schemes; $|\gamma|$: the number of attributes in access policy tree $\mathcal{T}$; ϑ: the number of attributes needed to satisfy policy $\mathcal{T}$; N_R: the number of data requesters.

5.2.2 Communication Overhead

Each ciphertext is composed of two parts: $[m_i] = \{T_i, T_i'\}$. It is highly related to the length of n^2, which has $2\mathcal{L}(n)$ bits. Hence, it has to transmit $4\mathcal{L}(n)$ bits for each ciphertext. Here, we set the bit length of each element in ABE to be $\mathcal{L}_\Delta$.

Further, we summarize the communication overhead in each scheme and compare with existing work with the assumption of N pieces of provided data, which is shown in Table 3. Notably, the communication overhead in Data Analysis (e.g., Add., Sub. and Mul., etc.) can be amortized over multi-user accesses, while the communication cost during Data Access is needed for each authorized DR. Moreover, the communication overhead caused by data provision can also be amortized by other data processing, which is not counted in Table 3. We can observe that the communication overhead in *Addition* and *Subtraction* does not rely on the number of provided data, while the communication overhead in *Multiplication* is proportional to the number of provided data. But the cost in other schemes does not vary much as it has only one or two data inputs. From the comparisons, we can find that the communication cost during data analysis in our new scheme is independent of the number of DRs, which saves the cost for multi-user access. In general, the communication cost is reasonable and our schemes are suitable for various applications.

TABLE 3
Communication Overhead of Each Scheme

Schemes	Communication overhead of our work	Communication Overhead in [18]
Add	$8\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	$8\mathcal{L}(n)N_R$
Sub.	$8\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	$8\mathcal{L}(n)N_R$
Mul.	$4(N + 1)\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	$4(N + 3)\mathcal{L}(n)N_R$
Sign	$9\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	$12\mathcal{L}(n)N_R$
Abs.	$13\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	–
Comp.	$9\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	$12\mathcal{L}(n)N_R$
Equal	$18\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta$	$24\mathcal{L}(n)N_R$
Vari.	–	$4(2N + 1)\mathcal{L}(n)N_R$
Data Access	$N_R(4\mathcal{L}(n) + (\|\gamma\| + 1)\mathcal{L}_\Delta)$	$4\mathcal{L}(n)N_R$

Note: $\mathcal{L}_\Delta$: bit length of elements in ABE; N_R: the number of data requesters.

5.2.3 Experimental Results

The efficiency of HRES has been presented in [18]. Hence, we focused on the performance evaluation of the proposed seven operation schemes. We implemented the proposed seven computing operation functions and tested their performances to check with aforementioned theoretical analysis. The evaluations are performed on a laptop with Intel Core i5-3337U CPU 1.8 GHz and 8 GB RAM with Java Pairing-Based Cryptography library (jPBC). To achieve better accuracy, we tested each algorithm 500 times and reported the average value of all testing results. Unless particularly specified, the parameters in our tests are set as the default values listed in Table 4.

Note: In our simulations, we set $\mathcal{L}(ck_1) = \mathcal{L}(ck_2) = 250$ bits and employ the curve of TYPE A in jPBC (http://gas.dia.unisa.it/projects/jpbc/docs/ecpg.html#TypeA). If higher security is desired, TYPE E can be adopted to support keys with longer length $\mathcal{L}(ck_1) = \mathcal{L}(ck_2) = 500$ bits. In our test machine, one pairing can be computed in approximately 9.8 milliseconds (ms).

In our proposed schemes, we adopt HRES to achieve the privacy-preserving data processing, while ABE is used to realize the data access control. As they are influenced by different parameters, we analyze the efficiency of data processing and access control separately. The CP in Step 4 and DSP in Step 5 encrypt the partial decryption key using ABE and

TABLE 4
Parameter Settings

Parameter	Value (bits)
$\mathcal{L}(n)$	1024
$\mathcal{L}(m_i)$	$\mathcal{L}(n)/4$
$\mathcal{L}(sk_i) = \mathcal{L}(sk_j) = \mathcal{L}(sk_{DSP}) = \mathcal{L}(sk_{CP})$	500
$\mathcal{L}(ck_1) = \mathcal{L}(ck_2)$	250
$\mathcal{L}(r_*) = \mathcal{L}(c_*)$	500
$\mathcal{L}(R)$	$(\mathcal{L}(n)/4) - 10$

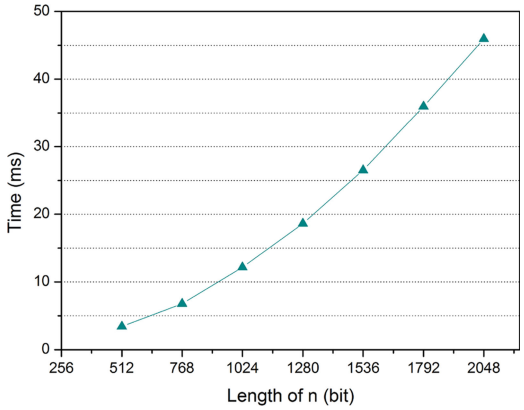

Fig. 3. Operation time of DPs with different length of n.

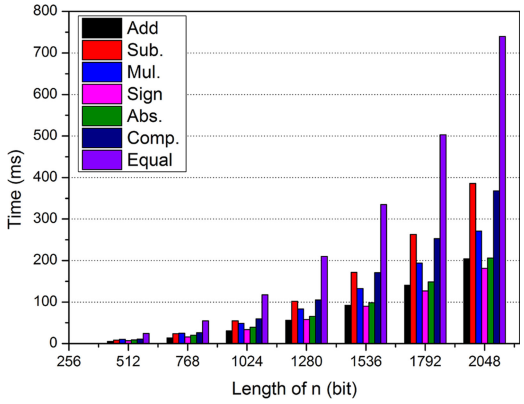

Fig. 4. Operation time of DSP in Step 3 with different length of n.

the DR decrypts to obtain the raw key with ABE in Step 6, the computation time of which are all presented in *Test 4*. Other data processing computation costs introduced by the ABE algorithm are simulated in *Tests 1*, *2* and *3*.

1) Efficiency of Data Processing

Test 1: Influence of the length of n on data processing

We presented the computation time of four involved entities: DP, DSP, CP, and DR respectively. The DP only involves in Step 2, which is similar in all schemes. We directly tested its computation cost with different length of n as shown in Fig. 3. We can observe that it is efficient and acceptable for DP with constrained resources.

The computation costs of DSP include two parts: Step 3 and Step 5. For a clearer presentation, we do not combine them, but present them in Figs. 4 and 6 respectively. We can observe that the cost grows with the increase of the bit length of n. In addition, the *Equality Test* is the most time-consuming, which needs about 750 ms when n achieves 2048 bits, while other algorithms take less than 400 ms. The DSP needs to process the *Comparing* twice to complete *Equality Test*. The performance evaluation of which is consistent with our analysis in Section 5.2.2. The computation cost of the DSP in Step 5 is shown in Fig. 6. Generally, our proposed schemes are acceptable for the cloud service provider, DSP.

From Fig. 5 that shows the computation cost of the CP, the *Equality Test* is also the most time-consuming while others take merely about 100 ms. As shown in Fig. 7, the computation cost of the DR is much less than those of two servers. Even when the bit length of n reaches 2048 bits, it still only needs about 25 ms to complete the computation.

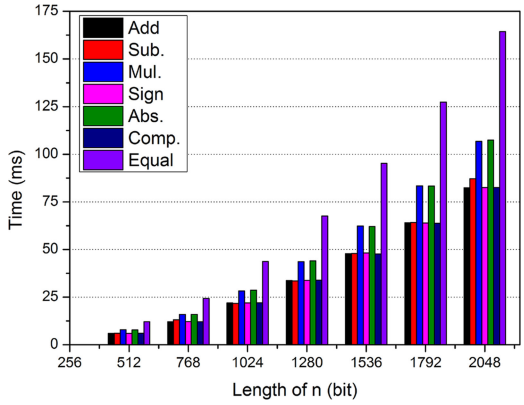

Fig. 5. Operation time of CP with different length of n.

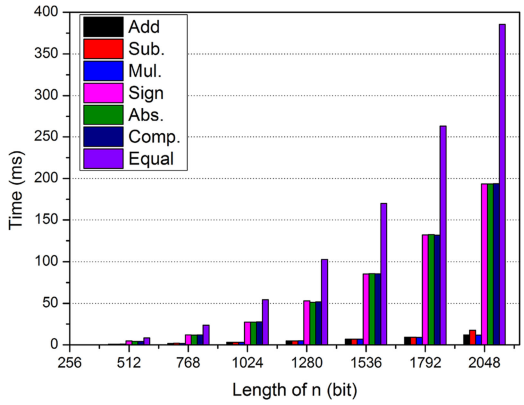

Fig. 6. Operation time of DSP in Step 5 with different length of n.

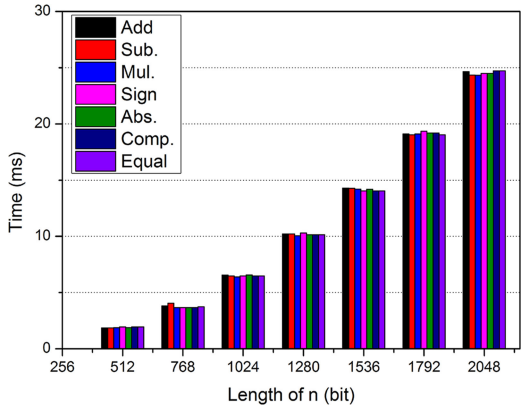

Fig. 7. Operation time of DR with different length of n.

In general, the above tests prove that most computation costs are undertaken by the DSP and the CP. The cloud users do not have much computation overhead. This result shows the practical advantage of the proposed schemes in the usage of mobile environments.

2) Flexibility of Addition, Subtraction and Multiplication

The operations of DSP include Step 3 and Step 5, the time of which are marked separately as DSP(1) and DSP(2) in Figs. 8, 9, and 10.

Test 2: Performance of Addition and Subtraction with a large number of provided data

In this experiment, we tested the performance of three kinds of system entities (i.e., DSP, CP and DR) in *Addition* with different numbers of provided data ($N = 10, 10^2, 10^3, 10^4, 10^5$). As shown in Fig. 8, the computation cost of the

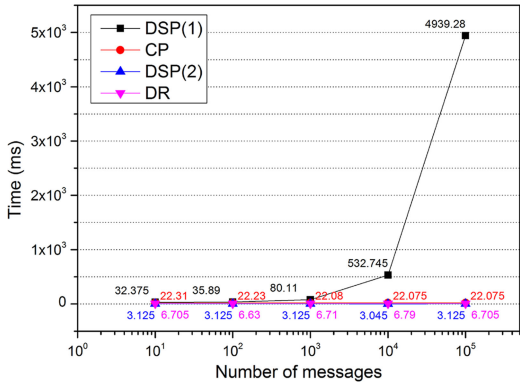

Fig. 8. Operation time of each entity in *Addition* with different number of DPs.

DSP in Step 3 increases with the number of the provided data, while it does not influence other steps. Even when the number of provided data reaches 10^4, the DSP only takes about 530 ms. It takes almost the same time for the DR to complete the computation with different numbers of provided pieces of data. This fact implies that our schemes can deal with a great number of data for addition efficiently.

We assume that the subtraction formula is $(\sum_{i=1}^{W} m_i - \sum_{i=W+1}^{N} m_i)$ with $W = N/2$, which means that half of the provided data is subtracted from the sum of another half. Fig. 9 shows its performance, which is similar to that of *Addition* and is efficient to support a big number of DPs.

In general, the scheme can flexibly be used in various situations with different number of data providers.

Test 3: Performance of Multiplication with a large number of provided data

Different from *Addition* and *Subtraction*, the *Multiplication* is time-consuming and communication-consuming. It is not flexible and possible to support the computation of a huge number of provided data. Thus, we only tested it with limited numbers $(N = 100,\ 200,\ 300,\ 400,\ 500,\ 600,\ 700,\ 800)$. Moreover, the original data length is set as $\mathcal{L}(n)/N$.

As shown in Fig. 10, it takes about 15 seconds to finish the computation in Step 3 for the DSP when the number of provided messages achieves 800. However, the data numbers do not influence the computation cost of the DR and no extra overhead is introduced, which merely costs about 6.7 ms.

3) Efficiency of Attribute-Based Encryption

The non-colluding servers and the DRs employ ABE to realize the flexible access control over the result of data

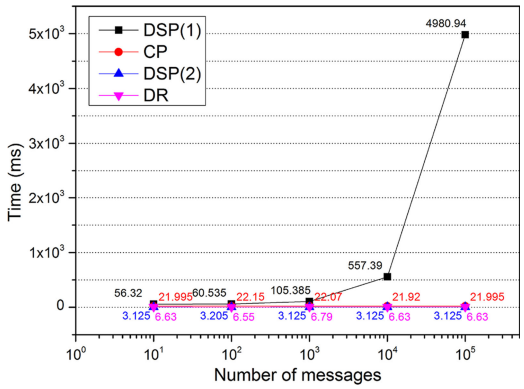

Fig. 9. Operation time of each entity in *Subtraction* with different number of DPs.

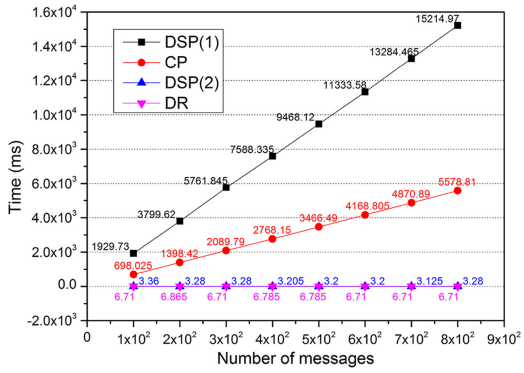

Fig. 10. Operation time of each entity in *Multiplication* with different number of DPs.

processing and the data retrieve, respectively. Here, we focus on the performance of ABE through one test.

Test 4: Performance of ABE with different numbers of attributes

In this experiment, we tested the performance of five steps of ABE with different number of attributes: System Setup (Setup_ABE), Encryption (Enc_ABE), Key Generation (KeyGen_ABE), Decryption (Dec_ABE) and its homomorphic computation (HE_ABE). The test was executed with the setting that all universal attributes are involved in encryption and that one attribute is needed to satisfy the policy tree. That is, $\mathcal{O}(|U|) = \vartheta$, which varies from 2 to 8 in our tests while $\mathcal{O}(|\gamma|) = 1$.

Fig. 11 shows the costs of all operations in ABE except the algorithm HE_ABE, as it only takes less than 1ms. By applying trust-based access control schemes [15], [16], [17], the decryption only involves one attribute and takes only 10 ms. We can observe that the computation cost of other algorithms is proportional to the number of attributes. Though employing ABE would cause high computation overhead, high security and fine-grained access control can be supported. Most computations are taken by cloud servers. In addition, the cloud servers only need to perform the setup and encryption once, which can be amortized by multiple accesses of DRs.

4) Experimental Comparison with Existing Work

Test 5: Performance comparison of Multiplication schemes in our work with an existing scheme in [18]

In this test, we compared the performance of *Multiplication* in our scheme with an existing scheme [18], as shown in Fig. 12. We obtained the total computation time of all

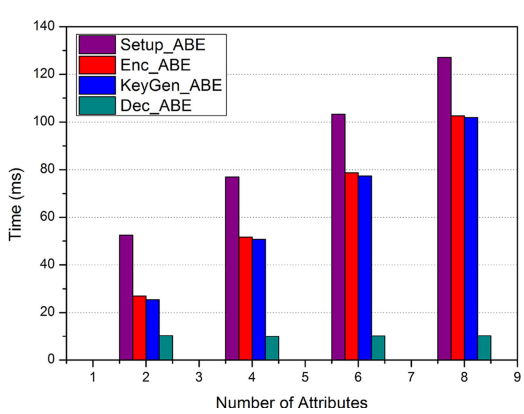

Fig. 11. Operation time of KPABE with different numbers of attributes.

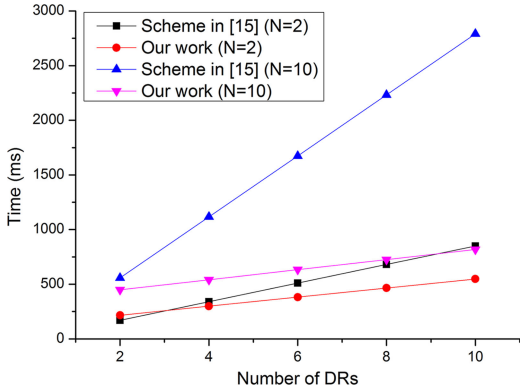

Fig. 12. Performance comparison of *Multiplication* schemes in our work and previous scheme [18].

involved entities with different number of DRs ($N_R = 2, 4, 6, 8, 10$). By comparing the performance of the two schemes with two DPs, we can see that our proposed scheme outperforms that in [18] when the number of DRs is over 3. Moreover, we compared the performance of two schemes with different number of DPs ($N = 2,\ 10$). By observing the simulation results, we find that our scheme shows much superiority in performance, especially for a big number of DPs.

5.3 Comparison with Existing work

Herein, we further compare our proposed scheme with some related work. With regard to secure data processing, HE-based schemes [5], [30] provide a feasible way to deal with the computation over encrypted data. But they have a serious problem because they are all single-user systems, which restricts the possibility of multi-user access control. SMC-based scheme [27] enables flexible access via secret sharing. However, they all introduce high computation overhead in big data processing, especially for multiplication. For a more intuitive comparison, we compare their communication cost and computation cost in multiplication over N pieces of data with our scheme in Table 5. The result shows the advantages of our scheme in terms of communication and computation efficiency. Moreover, no existing work overcomes the challenges to realize multiple computations over encrypted data and flexible access control over the processing result. Obviously, our work overcomes the current research challenges and effectively achieves flexible and secure computations over encrypted data with fine-grained access control.

6 Conclusion

In this paper, we proposed an efficient and secure scheme to achieve privacy-preserving data processing with ABE-based flexible access control. It can support seven basic operations and achieve fine-grained access control without the need of fully trusted cloud servers. Security analysis, performance evaluation and performance comparison with existing work further demonstrated that our scheme is efficient and effective with regard to big data processing operations. In the future, we are going to realize more operations, improve scheme efficiency and overcome latency by applying edge computing and pre-processing technologies. On the other hand, we will further explore significant applications of our scheme towards practical use.

TABLE 5
Comparison with Existing Work in Multiplication

	Communication cost (bits)	Computation cost
PHE-based [5]	$36 * N * \sigma$ ($\sigma = 1024$)	18^*N operations over Z_{2^σ} ($\sigma = 1024$)
SMC-based [27]	$480 * (N-1)$ (with 32-bit data)	3^N operations over $Z_{2^{32}}$
FHE-based [30]	0	N operations over Z_{2^σ} ($\sigma > 10^6$)
Our work	$(4N+3) * \sigma$ ($\sigma = 1024$)	$(4N+3)$ operations over Z_{2^σ} ($\sigma = 1024$)

Note: σ is the basic parameter and is different in each scheme; N: the number of provided data;

Acknowledgments

This work is sponsored by the National Key Research and Development Program of China (grant 2016YFB0800704), the NSFC (grants 61672410 and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), the Fundamental Research Funds for the Central Universities (grant JBG161509), the 111 project (grants B08038 and B16037), Academy of Finland (grant 308087), and the AXA Research Fund.

References

[1] A. Belle, R. Thiagarajan, S. Soroushmehr, F. Navidi, D. A. Beard, and K. Najarian, "Big data analytics in healthcare," *BioMed Res. Int.*, vol. 6, 2015, Art. no. 370194.
[2] J. J. Stephen, S. Savvides, R. Seidel, and P. Eugster, "Practical confidentiality preserving big data analysis," in *Proc. 6th USENIX Workshop Hot Topics Cloud Comput.*, 2014, pp. 10–10.
[3] B. Wang, M. Li, S. S. Chow, and H. Li, "A tale of two clouds: Computing on data encrypted under multiple keys," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2014, pp. 337–345.
[4] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.
[5] X. Liu, R. Choo, R. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Dependable Secure Comput.*, vol. PP, no. 99, 2016, doi: 10.1109/TDSC.2016.2536601.
[6] X. Liu, R. Deng, W. Ding, R. Lu, and B. Qin, "Privacy-preserving outsourced calculation on floating point numbers," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2513–2527, Nov. 2016.
[7] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *IACR Cryptology ePrint Archive*, vol. 2014, 2014, Art. no. 331.
[8] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *J. Netw. Computer Appl.*, vol. 42, pp. 120–134, 2014.
[9] A. Khedr and G. Gulak, "SecureMed: Secure medical computation using GPU-accelerated homomorphic encryption scheme," *IEEE J. Biomed. Health Inf.*, Jan. 2017, doi: 10.1109/JBHI.2017.2657458.
[10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innovations Theoretical Comput. Sci. Conf.*, 2012, pp. 309–325.
[11] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.
[12] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Adv. Cryptology–EUROCRYPT*, 2010, pp. 24–43.
[13] V. C. Hu, T. Grance, D. F. Ferraiolo, and D. R. Kuhn, "An access control scheme for big data processing," in *Proc. Int. Conf. Collaborative Comput.: Netw. Appl. Worksharing*, 2014, pp. 1–7.
[14] Z. Yan, W. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," *Future Generation Comput. Syst.*, vol. 62, pp. 175–189, 2015.

[15] C. Huang, Z. Yan, N. Li, and M. Wang, "Secure pervasive social communications based on trust in a distributed way," *IEEE Access*, vol. 4, pp. 9225–9238, 2016.
[16] Z. Yan, X. Li, M. Wang, and A. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 485–498, Jul.-Sep. 2015.
[17] Z. Yan, X. Li, and R. Kantola, "Controlling cloud data access based on reputation," *Mobile Netw. Appl.*, vol. 20, no. 6, pp. 828–839, 2015.
[18] W. Ding, Z. Yan, and R. H. Deng, "Encrypted data processing with homomorphic re-encryption," *Inf. Sci.*, vol. 409-410, pp. 35–55, 2017.
[19] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont, "Protecting and evaluating genomic privacy in medical tests and personalized medicine," in *Proc. 12th ACM Workshop Privacy Electron. Soc.*, 2013, pp. 95–106.
[20] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sens. Netw.*, vol. 5, no. 3, 2009, Art. no. 20.
[21] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 200–214.
[22] Q. Li, G. Cao, and T. La Porta, "Efficient and privacy-aware data aggregation in mobile sensing," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 2, pp. 115–129, Mar. 2014.
[23] Q. Li and G. Cao, "Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2013, pp. 60–81.
[24] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2013, pp. 111–125.
[25] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. 18th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2011, pp. 1–17.
[26] D. Bogdanov, R. Talviste, and J. Willemson, "Deploying secure multi-party computation for financial data analysis," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2012, pp. 57–64.
[27] L. Kamm and J. Willemson, "Secure floating point arithmetic and private satellite collision analysis," *Int. J. Inf. Secur.*, vol. 14, no. 6, pp. 531–548, 2015.
[28] D. Bogdanov, "Sharemind: Programmable secure computations with practical applications," PhD dissertation, University of Tartu, Estonia, 2013.
[29] J. H. Cheon, et al., "Batch fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2013, pp. 315–335.
[30] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Exploring the feasibility of fully homomorphic encryption," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 698–706, Mar. 2015.
[31] L. Morris. "Analysis of partially and fully homomorphic encryption," (2017). [Online]. Available: http://www.liammorris.com/crypto2/Homomorphic%20Encryption%20Paper.pdf
[32] X. Liu, R. H. Deng, Y. Yang, H. N. Tran, and S. Zhong, "Hybrid privacy-preserving clinical decision support system in fog–cloud computing," *Future Generation Comput. Syst.*, vol. 78, pp. 825–837, 2018.
[33] Z. Yan, W. Ding, and H. Zhu, "A scheme to manage encrypted data storage with deduplication in cloud," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2015, pp. 547–561.
[34] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2008, pp. 127–143.
[35] W. C. Garrison III, A. Shull, S. Myers, and A. J. Lee, "On the practicality of cryptographically enforcing dynamic access control policies in the cloud," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 819–838, doi: 10.1109/SP.2016.54.
[36] T. Zhu, W. Liu, and J. Song, "An efficient role based access control system for cloud computing," in *Proc. IEEE 11th Int. Conf. Comput. Inf. Technol.*, 2011, pp. 97–102.
[37] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.
[38] 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. Secur.*, 2006, pp. 89–98.
[39] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.
[40] 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.
[41] Z. Wan, J. E. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
[42] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Adv. Cryptology—EUROCRYPT*, 1999, pp. 223–238.
[43] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Proc. Adv. Cryptology-ASIACRYPT*, 2003, pp. 37–54.

Wenxiu Ding received the BEng degree in information security from Xidian University, Xi'an, China, in 2012. Now, she is working toward the PhD degree in information security in the School of Telecommunications Engineering, Xidian University. She was the research assistant in the School of Information Systems, Singapore Management University from 2015 to 2016. Her research interests include RFID authentication, privacy preservation, data mining and trust management.

Zheng Yan (M'06, SM'14) received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from Xi'an Jiaotong University, Xi'an, China, in 1994 and 1997, respectively, the second MEng 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 in electrical engineering from the Helsinki University of Technology (current Aalto University), Helsinki, Finland, in 2005 and 2007. She is currently a full professor with the Xidian University, Xi'an, China and a visiting professor and an academy research fellow with the Aalto University, Espoo, Finland. Her research interests include trust, security and privacy, as well as data mining. She is serving as an associate editor of the *IEEE Internet of Things Journal*, the *IEEE Access*, the *Information Sciences*, the *Information Fusion*, the *Journal of Network and Computer Applications*, the *Soft Computing*, the *Security and Communication Networks*, etc. twelve journals. She serves as an organization and program committee member for numerous international conferences. She is a senior member of the IEEE.

Robert H. Deng is AXA chair professor of Cybersecurity and director of the Secure Mobile Centre, School of Information Systems, Singapore Management University. His research interests include the areas of data security and privacy, cloud security and Internet of Things security. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards and conference committees. These include the editorial boards of the *IEEE Security & Privacy Magazine*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Information Forensics and Security*, the *Journal of Computer Science and Technology*, and Steering Committee chair of the *ACM Asia Conference on Computer and Communications Security*.