Secure and Efficient Data Processing for Cloud Computing with Fine-Grained Access Control

Jingjing Wang[1], Hao Feng[1], Zheng Yu[1], Rongtao Liao[1], Shi Chen[1], and Ting Liang[2(✉)]

[1] State Grid Hubei Electric Power Co., Ltd. Information Communication Company, Shiyan, China
[2] School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China
liangting@whut.edu.cn

Abstract. Nowadays, with rapid development of cloud computing, many information systems are running on the cloud platform. However, the cloud servers are not fully trustworthy, for the purpose of privacy preserving, the users need to encrypt their data before uploading it to the cloud. However, this also brings challenges in utilizing the data. Generally speaking, several desirable properties should be achieved for data processing on the cloud platform. First, the cloud servers should be able to perform computations on the encrypted data without learning users' sensitive information. Second, fine-grained access control should be enforced on the computed results. Third, flexibility requires that the identities who can access the computed results should be unknown when these results are generated. Fourth, the scheme should have low overheads on computation and communication. To the best of our knowledge, most of the existing schemes cannot satisfy these requirements simultaneously. In order to address this issue, we propose a secure and efficient privacy preserving data processing scheme for cloud computing with fine-grained access control, using a homomorphic proxy re-encryption scheme and an efficient attribute-based encryption scheme. Security analyses prove that it satisfies all the desirable security properties, and performance evaluation demonstrates that it is more efficient than the state-of-the-art schemes targeting similar problems. In particular, the size of ciphertexts and the decryption time for the computed results are constant in our scheme, regardless the access structure. Therefore, our scheme contributes to a more practical data processing scheme for the cloud platform with fine-grained access control.

1 Introduction

Thanks to the appealing properties, such as reliability, mobility and cost saving, cloud computing has attracted great attentions nowadays and many information systems are running on the cloud platform. For example, wearable devices

B. Arief et al. (Eds.): SocialSec 2023, LNCS 14097, pp. 187–202, 2023.
https://doi.org/10.1007/978-981-99-5177-2_11

and smart medical services are widely deployed, providing great convenience for healthcare. In this application, cloud computing can help to reduce the heavy burden of edge devices [16]. In the era of big data, the storage and processing of data in the smart grid also need the assistance of cloud computing [17].

However, privacy preserving is a crucial requirement for most cloud-based applications, as the cloud servers are not fully trustworthy in many circumstances [11]. If users' sensitive information is leaked, their privacy will be violated, and sometimes it can cause catastrophic consequences. Recently, many countries have issued laws and regulations for data protection. One feasible solution is that the users encrypt their data before uploading it to the cloud platform, but this also brings challenges in utilizing the data. To address this issue, many research efforts have been devoted in privacy preserving computations, where homomorphic encryption and multiparty computation are two popular cryptographic techniques. In both approaches, the encrypted data can be processed or analyzed without leaking its underneath plaintexts [8,18,20].

In some scenarios, not only users' sensitive information but also the processing results should be protected. In the literature, a few researchers have also investigated this issue and some solutions have been proposed with fine-grained access control on the computed results. However, most of these schemes are either inflexible or inefficient. For example, some schemes [23] require that the identities who can access the computed results are known by the time these results are generated, making them less versatile in real-world applications. For example in smart grid, the staffs responsible for data management need to rotate frequently, hence some staffs will take this position after the computed results are generated. In some other schemes [5,21], the computational complexity in decrypting the computed results is relatively high, i.e. the size of ciphertexts and the decryption time are proportional to the number of attributes.

To the best of our knowledge, very few schemes can satisfy the above desirable features simultaneously. In order to fulfil these requirements, we propose a secure and efficient data processing scheme for cloud computing with fine-grained access control, using a homomorphic proxy re-encryption scheme and an efficient attribute-based encryption scheme as the main building blocks. Our contributions can be summarized as follows:

- Our scheme not only allows users' encrypted data to be processed without leaking their sensitive information, but also achieves fine-grained access control for the computed results in a flexible way, i.e. the identities who can access the computed results can be configured after these results are generated.
- Our scheme is more efficient than the state-of-the-art schemes that satisfy similar properties. In particular, the size of ciphertexts and the decryption time for the computed results can be made constant in our scheme, regardless the access structure.

2 Related Works

Cloud computing provides sufficient storage space and computing power, and it has become an important infrastructure for information systems. Privacy

preserving and access control are two security issues that need to be considered for processing and utilizing the data in the cloud platform. In this section, we review some existing works on privacy preserving data processing and fine-grained access control.

Multiparty computation allows several participants to jointly generate the outputs for some function without leaking each individual input, and it has already been demonstrated that any Turing computable function can be evaluated by MPC [15]. For example, it has already been designed for various different tasks, such as federated learning [2], computation of biomedical data [12], and avoidance for satellite collision [13]. Therefore, this technique can be used for privacy preserving data processing in cloud computing. However, for general purpose MPC, it is still impractical for large scale applications due to the heavy overheads in computation and communication.

Homomorphic encryption enables to perform computations on the encrypted data, and the same effect applies as if the operations are performed on the corresponding plaintexts. This technique can be further divided into two categories: fully homomorphic encryption [9] and partial homomorphic encryption [4]. The first type is more versatile, i.e. it supports both addition and multiplication. However, its performance still cannot meet the demands in real-world applications due to the large key size and high storage overheads. The second type only supports either addition or multiplication. But it is more efficient and many schemes have already been deployed in large scale applications. For example, it has been used to design secure e-voting, privacy preserving data aggregation and federated learning.

Fine-grained access control requires that the information can only be accessed by the designated recipient whose attributes satisfy the access control policy. In cryptography, there are two approaches to guarantee this property: proxy re-encryption (PRE) and attribute-based encryption (ABE). The first type allows the ciphertext encrypted under one entity's public key to be transformed into a different ciphertext encrypted under another entity's public key. In [23], Zhang et al. used PRE to propose a privacy preserving data aggregation scheme for smart grid with fine-grained access control. However, it requires that the identities who can access the computed results are known in advance before these results being generated. The second type also solves this issue, and it is more flexible [1, 10], but it suffers some limitations. First, most existing ABE schemes do not have the additive homomorphic property that is required for privacy preserving computations. And second, their computational overheads are relatively high when the access structure is complex, because the size of ciphertexts and the decryption time are proportional to the number of attributes [5,21].

Recall that our purpose is to design a secure and efficient data processing scheme with privacy preserving and fine-grained access control. Based on the above analyses, it is non-trivial to achieve the purpose just using either PRE or ABE. Instead, our design integrates a homomorphic PRE scheme and a novel ABE scheme. The advantage is that it not only satisfies the desirable security

features, but also is more efficient than the state-of-the-art schemes with similar properties.

3 Preliminaries

In this section, we briefly review some cryptographic primitives that will be used to design our proposed scheme.

3.1 A Homomorphic Re-encryption Scheme

In [4], Bresson, Catalano and Pointcheval have proposed an encryption scheme, called the BCP scheme, with the additive homomorphic property. Compared with Paillier encryption, the BCP scheme has two unique features. First, it contains two trapdoors: the master trapdoor can decrypt any ciphertext, while a user's trapdoor can only decrypt the ciphertexts under her public key. Second, its mathematical structure contains a finite cyclic group of quadratic residues in which the discrete logarithm assumption holds, and this feature enables it to be extended into a proxy re-encryption scheme. The BCP scheme works as follows:

- **Initialization.** Given the security parameter κ, one chooses two large safe primes p, q, such that $p = 2p' + 1, q = 2q' + 1$ where p', q' are also primes. Denote $n = pq$ and $\lambda = p'q'$. Let $\mathbb{G} = \mathsf{QR}_{n^2}$ be the cyclic group of quadratic residues modulo n^2. We have $\mathsf{ord}(\mathbb{G}) = n\lambda$. Then, one randomly chooses a value $\alpha \in \mathbb{Z}^*_{n^2}$ and compute $g = \alpha^2 \bmod n^2$. Now, g is a random element in $\mathbb{G}$ and it has maximal order with overwhelming probability. Finally, the system parameters (n, g) are made public.
- **Key generation.** The user randomly chooses a value $a \in [1, \mathsf{ord}(\mathbb{G})]$ and sets $h = g^a \bmod n^2$. Her public key is h and her private key is a.
- **Encryption.** Given a message $m \in \mathbb{Z}_n$, one randomly chooses a value $r \in [1, \mathsf{ord}(\mathbb{G})]$ and computes the ciphertext $C = (A, B)$ as:

$$A = g^r \bmod n^2 \qquad B = h^r(1+n)^m \bmod n^2$$

- **Decryption.** The user can decrypt the ciphertext using her private key as:

$$m = \frac{B/A^a - 1 \bmod n^2}{n}$$

Some Remarks. Note that the value $\mathsf{ord}(\mathbb{G})$ should be unknown in the key generation and encryption algorithms, because this value reveals the master trapdoor and it enables one to decrypt any ciphertext. Instead, one can randomly select a value in $[1, n^2/4)$ instead of $[1, \mathsf{ord}(\mathbb{G})]$. The statistical distance between these two sets is $O(n^{-1/2})$, which is indistinguishable for any probabilistic polynomial time (PPT) adversary. In the rest of the paper, we will ignore this issue and simply use $\mathsf{ord}(\mathbb{G})$ in the description. Besides, we assume that all computations are modular n^2 unless otherwise stated.

In [7], Ding et al. have extended the BCP scheme into a proxy re-encryption scheme, such that a ciphertext can not only be decrypted, but also be transformed into a different ciphertext under another public key. The proxy re-encryption process is implemented by two proxies, but it also can be done in the distributed fashion by arbitrary number of proxies as shown in [22]. Ding's scheme works as follows:

- **Initialization.** Same as in the BCP scheme.
- **Key generation.** Two proxies generate their key pairs separately. The first proxy randomly selects $a \in [1, \mathsf{ord}(\mathbb{G})]$ and sets $h_a = g^a$. The second proxy randomly selects $b \in [1, \mathsf{ord}(\mathbb{G})]$ and sets $h_b = g^b$. The public keys (h_a, h_b) are published. Moreover, they negotiate a Diffie-Hellman key as $h = {h_a}^b = {h_b}^a = g^{ab}$, and publish h.
- **Encryption.** Same as in the BCP scheme. We assume that the message is encrypted under the public key h.
- **Proxy re-encryption.** Denote $\mathsf{H} : \{0,1\}^* \rightarrow [1, \mathsf{ord}(\mathbb{G})]$ as some cryptographic hash function. To re-encrypt a ciphertext $(A, B) = (g^r, h^r(1+n)^m)$ under the public key $h = g^{ab}$ to a different ciphertext under another public key $\hat{h} = g^x$, the first proxy computes $\sigma_1 = \mathsf{H}(\hat{h}^a)$ and $(A', B') = (A^a g^{\sigma_1}, B)$, and sends the result to the other proxy. Then, the second proxy computes $\sigma_2 = \mathsf{H}(\hat{h}^b)$ and $(A'', B'') = ({A'}^b g^{\sigma_2}, B')$, and sends the result to the designated recipient with public key $\hat{h}$.
- **Decryption.** This recipient can decrypt the ciphertext (A'', B'') by computing $\sigma_1 = \mathsf{H}({h_a}^x)$, $\sigma_2 = \mathsf{H}({h_b}^x)$ and

$$m = \frac{B'' \cdot {h_b}^{\sigma_1} \cdot g^{\sigma_2} / A'' - 1 \bmod n^2}{n}$$

3.2 An Efficient ABE Scheme

In [14], Li et al. have introduced a novel ABE scheme with AND-gate access structure. Compared with the traditional ABE schemes in which the decryption costs increase linearly with the number of attributes, the size of ciphertext and the number of bilinear pairing operations in Li's scheme remain constant in the decryption process. This scheme consists of four algorithms: *Setup*, *Encrypt*, *KeyGen*, and *Decrypt*, and it works as follows.

Setup. It takes as input the security parameter κ, and outputs the public parameters and system keys. The key generation center (KGC) first chooses two finite cyclic groups G_1, G_2, both with order p. g is denoted as the generator of G_1 and e is a bilinear map: $G_1 \times G_1 \rightarrow G_2$. The system parameters (e, g, p, G_1, G_2) are made public. Then, the KGC randomly picks $3n$ elements $h_1, h_2, \cdots, h_{3n}$ in G_1, where $n = |U|$ is denoted as the number of attributes in the system. h_i, h_{n+i} and h_{2n+i} denote three types of attributes: positive, negative and wildcard. Finally, the KGC randomly picks $\alpha, a \in \mathbb{Z}_p$ and calculates $Y = e(g,g)^\alpha$ and g^a. Now, the public key is $PK = (e, g, Y, g^a, h_1, h_2, \cdots, h_{3n})$, and the master private key is $MK = (g^\alpha, a)$.

KeyGen. It takes as input the user's attribute set S, and outputs the private key for this user. For each $i \in U \wedge i \in S$, the KGC sets the tag $i' = +i$. For each $i \in U \wedge i \notin S$, the KGC sets the tag $i' = -i$. Note that the attributes not in S are considered as negative states. The KGC randomly chooses $r, c \in \mathbb{Z}_p$ and sets $L' = c$. It then computes $D = g^{-r}, L = g^{-ar}$. For each attribute $i \in U$, it sets $D_i = {h_i}^r$ if $i' = +i$; and it sets $D_i = {h_{n+i}}^r$ if $i' = -i$. For each $i \in U$, it computes $F_i = {h_{2n+i}}^r$. The KGC chooses a random value $j \in U$ and computes ${D_j}' = g^{\alpha/(a+c)} \cdot D_j$, ${F_j}' = g^{\alpha/(a+c)} \cdot F_j$. As follows, the values D_j, F_j are replaced by D'_j, F'_j. Note that for any PPT adversary, such a substitution is indistinguishable. Finally, the KGC outputs $SK = (D, L, L', \langle D_i, F_i \rangle \mid i \in U)$.

Encrypt. It takes as input the access structure $W = \Lambda_{i \in I} i'$ and a message, and outputs a ciphertext. Note that AND-gate is employed in the access structure, while I denotes the attribute set of access policy and i' denotes the state of the attribute i. Each attribute has three different states: positive $(+i)$, negative $(-i)$, and wildcard. The last type means that the attribute is not mentioned in the access structure. To encrypt a message M, one first randomly picks $s \in \mathbb{Z}_p$ and calculates $C = M \cdot Y^s, C_1 = g^s$, and $C_2 = g^{as}$. Then, for each $i \in I \wedge i' = +i$, it sets $H_i = h_i$. For each $i \in I \wedge i' = -i$, it sets $H_i = h_{n+i}$. For each $i \in U \wedge i \notin I$, it sets $H_i = h_{2n+i}$. Finally, it computes $C_3 = (\prod_{i \in U} H_i)^s$. The ciphertext is $CT = (W, C, C_1, C_2, C_3)$.

Decrypt. It takes as input a ciphertext and the user's secret key, and outputs $\perp$ or a plaintext. If the set S does not satisfy W, it outputs the symbol $\perp$. Otherwise, for each $i \in I$: if $i' = +i \wedge i \in S$, it computes $A_1 = \prod D_i$; if $i' = -i \wedge i \notin S$, it computes $A_2 = \prod D_i$. For each $i \notin I$, it computes $A_3 = \prod_{i \in U \setminus I} F_i$. Then, it calculates

$$K = e(A_1 \cdot A_2 \cdot A_3, {C_1}^{L'} \cdot C_2) \cdot e(D^{L'} \cdot L, C_3) = e(g, g)^{\alpha s}$$

Finally, the plaintext M can be derived by computing $M = C/K$.

4 Models and Definitions

In this section, we outline the models and definitions, including the system model, the communication model, the adversary model, the security requirements, and the security assumptions.

4.1 System Model

There are five types of entities in our proposed scheme: authority, data service provider (DSP), computation party (CP), data provider (DP), and data requester (DR). The system model is shown in Fig. 1:

- **Authority**: Its main responsibility is to initialize the system parameters and generate secret keys for the users.

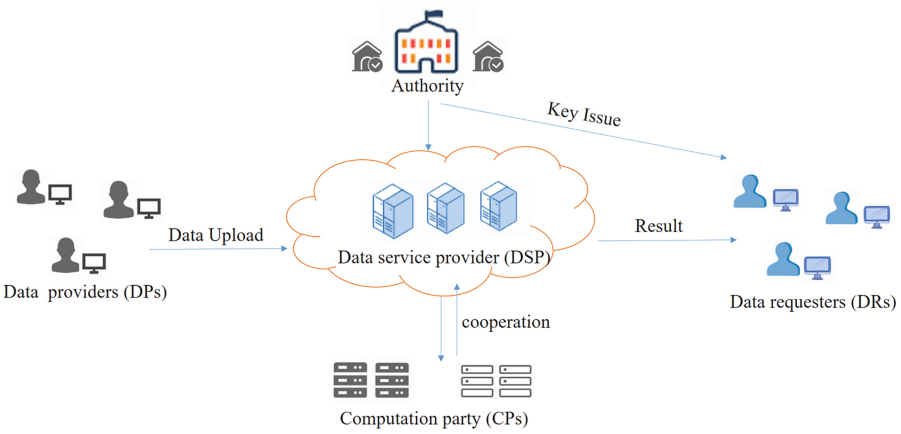


Fig. 1. The System Architecture

- **DSP**: It takes in charge of data storage, data sharing and data processing. For example, it stores the encrypted data uploaded by the DPs, performs data processing, collaborates with the CP for proxy re-encryption, and outputs the computed results to the DRs. The DSP is normally run by some cloud platform.
- **CP**: It is responsible for enforcing fine-grained access control on the computed results, collaborating with the DSP to perform proxy re-encryption.
- **DP**: The data providers are the data owners. They encrypt their data before uploading it to the cloud platform, where the uploaded data can be used for data analysis.
- **DR**: The data requester is a consumer of the computed results, subject to proper access control. In other words, if its attributes satisfy the access structure, it can decrypt the computed results. Otherwise, it learns no information of the computed results.

4.2 Communication Model

We assume that secure channels exist between the authority and the DRs as well as between the DSP and the CP. All other communications are assumed to be exchanged through authenticated channels. The adversary can neither intercept nor tamper with the messages transmitted through the secure channels. And in authenticated channels, the authenticity and integrity of the received messages can be verified. Note that the assumption of these channels enables us to focus on the protocol design without digging into low-level technical details, and these channels can be implemented by standard cryptographic primitives, such as encryption and digital signature.

4.3 Adversary Model

In our proposed scheme, the authority is assumed to be fully trustworthy. All other entities are assumed to be semi-honest, i.e. they will follow the protocol but be curious to learn information beyond their authorization. Besides, we assume that the DSP and the CP will not collude, e.g. parties with conflicting interests can be chosen. And an adversary $\mathcal{A}$ with the following capabilities is considered.

- $\mathcal{A}$ can eavesdrop the exchanged messages on the authenticated channels, but it cannot eavesdrop on the secure channels.
- $\mathcal{A}$ may compromise the DSP or the CP, but not both, with the purpose of learning DP's sensitive information or the computed results.
- $\mathcal{A}$ may compromise some DRs, trying to combine their attributes to form a larger set so that it can access the computed results that none of these DRs is authorized.

To prevent trivial break of our proposed scheme, we assume that $\mathcal{A}$ will not compromise a DP to learn its uploaded data, and $\mathcal{A}$ will not compromise a DR to obtain its decryption privilege.

Some Remarks. Note that in real-world applications, the authority's power can be distributed among multiple parties and the behavior of all entities can be verified by zero-knowledge proofs. Hence, there is no need to assume that the authority is fully trustworthy and all entities are semi-honest. The assumption here is only to simplify the description of our proposed protocol.

4.4 Security Requirements

The following security requirements are considered in our proposed scheme.

- **Correctness.** If all participants honestly follow the protocol, the uploaded encrypted data can be processed in the privacy preserving way and the computed results can only be decrypted by the designated recipient.
- **Privacy.** $\mathcal{A}$ can neither learn the data stored in the cloud platform nor the computed results output by the CP.
- **Fine-grained access control.** Only the designated recipient whose attributes satisfy the access structure can decrypt the computed results.
- **Flexibility.** The identities who can access the computed results should be unknown when these results are generated, i.e. some parties can register after the computed results are generated and still be able to decrypt them.
- **Collusion resistance.** The DRs cannot collude to gain more decryption privilege by combining their attributes.

4.5 Security Assumptions

DDH Assumption over $Z_{n^2}^*$ [4]: Given two large safe primes p, q, and $n = pq$. $\mathbb{G}$ is denoted as the cyclic group of quadratic residues modulo n^2, and g is the generator of $\mathbb{G}$. $x, y \in [1, \mathsf{ord}(\mathbb{G})]$ are randomly selected and $X = g^x$, $Y = g^y$.

For every PPT adversary $\mathcal{A}$, it cannot distinguish the two elements $Z_0 = g^z$, $Z_1 = g^{xy}$ with non-negligible advantage. This statement can be expressed as:

$$\mathsf{Pr}[\mathcal{A}(g, X, Y, Z_0) = 1] - \mathsf{Pr}[\mathcal{A}(g, X, Y, Z_1) = 1] \leq \varepsilon$$

where the probability is taken over the random choice of g in $\mathbb{G}$, the random choice of $z \in [1, \mathsf{ord}(\mathbb{G})]$, and the random bits used by $\mathcal{A}$.

l-BDHE Assumption [3]: Given a bilinear map $e : G \times G \rightarrow G_1$, where both G and G_1 are finite cyclic groups with prime order p. g, h are two generators of G. Given a vector of $2l+1$ elements $(h, g, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^l)}, g^{(\alpha^{l+2})}, ..., g^{(\alpha^{2l})})$, for every PPT adversary $\mathcal{A}$, it cannot compute the value $e(g,h)^{\alpha^{l+1}} \in G_1$ with non-negligible probability. This statement can be expressed as:

$$\mathsf{Pr}[\mathcal{A}(h, g, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^l)}, g^{(\alpha^{l+2})}, ..., g^{(\alpha^{2l})}) = e(g^{(\alpha^{l+1})}, h)] \leq \varepsilon$$

where the probability is taken over the random choice of generators g, h in G, the random choice of $\alpha \in Z_p$, and the random bits used by $\mathcal{A}$.

5 The Proposed Scheme

In this section, we describe our proposed privacy-preserving data processing scheme with fine-grained access control. We first give a high-level overview of the scheme and then present its technical details.

5.1 An Overview

Our proposed scheme consists of the following seven algorithms:

- **System setup.** Given the security parameter, the authority initializes both the PRE scheme and the ABE scheme. Moreover, it selects a cryptographic hash function that maps from the plaintext space of ABE scheme to the plaintext space of the PRE scheme. Note that this step is crucial as the plaintext spaces for these two encryption schemes are not compatible.
- **Key generation.** The DSP and the CP each generates a key pair in the PRE scheme. They also negotiate a Diffie-Hellman key as the system-wide public key. Moreover, each DR registers with the authority, and receives her private key.
- **Encryption.** Each DP can encrypt her data using the BCP scheme, and uploads it to the cloud platform.
- **Data processing.** The DSP can perform data analysis and data mining using the stored data. Note that all operations are done on encrypted data without disclosing DPs' privacy.
- **Proxy re-encryption I.** Once the computed results are generated, the DSP first performs partial decryption as well as re-encryption on the computed results.

- **Proxy re-encryption II.** The CP continues to perform partial decryption and re-encryption on the computed results. In the output, the computed result have been transformed from a ciphertext in the BCP scheme into a ciphertext in the ABE scheme.
- **Decryption.** Finally, only the designated recipients whose attributes satisfy the access structure can decrypt the computed results

5.2 The Details Scheme

System Setup. In this phase, the authority initializes the protocol and generates the public parameters. First, given the security parameter κ, the authority chooses two large safe primes p', q' and computes $n = p'q'$. $\mathbb{G}$ is denoted as the cyclic group of quadratic residues modulo n^2 and $\bar{g}$ is the generator of $\mathbb{G}$. Next, the authority chooses two finite cyclic groups G_1, G_2, both with prime order p. g is denoted as the generator of G_1 and e is a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. Then, the authority randomly picks $3k$ elements $h_1, h_2, \ldots, h_{3k}$ in G_1, where $k = |U|$ is denoted as the number of attributes in the system. h_i, h_{k+i} and h_{2k+i} denote three types of attributes: positive, negative and wildcard. As follows, the authority randomly picks $\alpha, a \in \mathbb{Z}_p$, and computes $Y = e(g, g)^{\alpha}$ and g^a. Finally, two cryptographic hash functions are selected: $\mathsf{H} : \{0,1\}^* \rightarrow [1, \mathsf{ord}(\mathbb{G})]$, $\mathsf{H}' : G_2 \rightarrow \mathbb{Z}_n$. It publishes the public parameters $PK = (\bar{g}, n, \mathbb{G}, e, g, G_1, G_2, Y, g^a, h_1, h_2, \ldots, h_{3k}, \mathsf{H}, \mathsf{H}')$ and keeps the master secret key $MK = (g^{\alpha}, a)$ private.

Key Generation. Each entity generates its key pairs. For example, the DSP randomly chooses a value $x \in [1, \mathsf{ord}(\mathbb{G})]$ as its secret key and computes the public key $pk_{DSP} = \bar{g}^x$. The CP randomly chooses a value $y \in [1, \mathsf{ord}(\mathbb{G})]$ as its secret key and computes the public key $pk_{CP} = \bar{g}^y$. Afterwards, they negotiate a Diffie-Hellman key $\bar{h} = pk_{DSP}{}^{sk_{CP}} = pk_{CP}{}^{sk_{DSP}} = \bar{g}^{xy}$. Each DR registers with the authority to obtain her private key. Suppose a particular DR is with the attribute set S. For each $i \in U \wedge i \in S$ the authority sets the tag $i' = +i$; for each $i \in U \wedge i \notin S$ the authority sets the tag $i' = -i$. Then, the authority randomly chooses $r, c \in \mathbb{Z}_p$ and sets $L' = c$. It then computes $D = g^{-r}$ and $L = g^{-ar}$. For each $i \in U$, it sets $D_i = {h_i}^r$ if $i' = +i$, and it sets $D_i = {h_{k+i}}^r$ if $i' = -i$. For each $i \in U$, it computes $F_i = {h_{2k+i}}^r$. The authority then randomly chooses a value $j \in U$ and computes ${D_j}' = g^{\alpha/(a+c)} \cdot D_j$ and ${F_j}' = g^{\alpha/(a+c)} \cdot F_j$. As follows, the values D_j, F_j are replaced by ${D_j}', {F_j}'$. Finally, the authority publishes the public keys $pk_{DSP}, pk_{CP}, \bar{h}$, and sends the private key SK to each DR through secure channels.

$$SK = (D = g^{-r}, L = g^{-ar}, L' = c, \langle D_i, F_i \rangle \mid i \in U)$$

Encryption. Each DP encrypts its data $m_i \in \mathbb{Z}_n$ and uploads the ciphertext $(\bar{g}^r, \bar{h}^r(1+n)^{m_i})$ to the cloud platform.

Data Processing. The DSP can process the encrypted data according to the specific requirement. Note that the BCP scheme already enjoys the additive

homomorphic encryption, and Ding et al. [7] have also introduced a toolkit of basic operations over ciphertexts. Suppose that the computed result is a ciphertext $(A, B) = (\bar{g}^r, \bar{h}^r(1+n)^m)$.

Proxy Re-encryption I. The DSP randomly selects $w_1 \in G_2$ and encrypts it as $CT_1 = (W, C, C_1, C_2, C_3)$, where $W = \Lambda_{i \in I} i'$ is the access structure for the designated recipients, $C = w_1 \cdot Y^{s_1}$, $C_1 = g^{s_1}$ and $C_2 = g^{as_1}$. For each $i \in I \wedge i' = +i$, it sets $H_i = h_i$. For each $i \in I \wedge i' = -i$, it sets $H_i = h_{k+i}$. For each $i \in U \wedge i \notin I$, it sets $H_i = h_{2k+i}$. And $C_3 = (\prod_{i \in U} H_i)^{s_1}$. The DSP then transforms the ciphertext (A, B) into $(A', B') = (\bar{g}^{xr}, h^r(1+n)^{m+\sigma_1})$, where $\sigma_1 = \mathsf{H}'(w_1)$, and sends (A', B') and CT_1 to the CP through a secure channel.

Proxy Re-encryption II. The CP randomly selects $w_2 \in G_2$ and encrypts it as $CT_2 = (W, C', C_1', C_2', C_3')$, where $C' = w_2 \cdot Y^{s_2}$, $C_1' = g^{s_2}$, $C_2' = g^{as_2}$ and $C_3' = (\prod_{i \in U} H_i)^{s_2}$. The CP then transforms the ciphertext (A', B') into $(A'', B'') = (\bar{g}^{xyr}, \bar{h}^r(1+n)^{m+\sigma_1+\sigma_2})$, where $\sigma_2 = \mathsf{H}'(w_2)$, and sends (A'', B''), CT_1 and CT_2 to the designated DR.

Decryption. The DR first computes $m' = m + \sigma_1 + \sigma_2$ by

$$m' = \frac{B''/A'' - 1 \bmod n^2}{n}$$

Then, it decrypts CT_1 and CT_2, obtaining w_1 and w_2, respectively. For example, to decrypt CT_1, the DR computes

$$w_1 = \frac{C}{e(A_1 \cdot A_2 \cdot A_3, C_1^{L'} \cdot C_2) \cdot e(D^{L'} \cdot L, C_3)}$$

where $A_1 = \prod D_i$ for each $i \in I$ where $i' = +i \wedge i \in S$, $A_2 = \prod D_i$ for each $i \in I$ where $i' = -i \wedge i \notin S$, and $A_3 = \prod_{i \in U \setminus I} F_i$. The ciphertext CT_2 can be decrypted similarly. Finally, The plaintext m can be derived by computing

$$m = m' - \mathsf{H}'(w_1) - \mathsf{H}'(w_2) \bmod n$$

6 Security Analyses

In this section, we prove that the proposed scheme achieves the desirable security properties, such as correctness, privacy, fine-grained access control, flexibility and collusion resistance.

Theorem 1. *The proposed scheme satisfies the correctness property.*

Proof. First, we prove that if the DSP and the CP are honest, the computed result can be correctly transformed into a ciphertext under some access structure. Second, we prove that the party whose attributes satisfy the access structure can decrypt the computed result. To see the first point, once receiving a ciphertext

$(A, B) = (\bar{g}^r, \bar{h}^r(1+n)^m)$, the DSP first partially decrypt it, and then uses a random value $\sigma_1 \in \mathbb{Z}_n$ to blind the plaintext as

$$(A', B') = (A^x, B \cdot (1+n)^{\sigma_1}) = (\bar{g}^{xr}, \bar{h}^r(1+n)^{m+\sigma_1})$$

Similarly, once the CP receives (A', B'), it will partially decrypt it, and uses another random value $\sigma_2 \in \mathbb{Z}_n$ to further blind the plaintext as

$$(A'', B'') = (A'^y, B' \cdot (1+n)^{\sigma_2}) = (\bar{g}^{xyr}, \bar{h}^r(1+n)^{m+\sigma_1+\sigma_2})$$

At this moment, anyone can derive $m' = m + \sigma_1 + \sigma_2$ as

$$m' = \frac{B''/A'' - 1 \bmod n^2}{n}$$

where $\sigma_1 = \mathsf{H}'(w_1), \sigma_2 = \mathsf{H}'(w_2)$, and w_1, w_2 are randomly chosen in G_2. Therefore, if a party can decrypt the ciphertexts for w_1, w_2, she can decrypt the computed result. Recall that w_1, w_2 are both encrypted using an ABE scheme according to some access structure. Therefore, the party whose attributes satisfy this access structure can decrypt these values. Therefore, our proposed scheme satisfies the correctness property.

Theorem 2. *The proposed scheme satisfies the privacy property.*

Proof. To prove that the adversary $\mathcal{A}$ cannot learn any information in our proposed scheme. We need to prove that neither $\mathcal{A}$ can learn information from DP's uploaded encrypted data, nor $\mathcal{A}$ can learn information from the transformed ciphertext. Recall that DP's uploaded data is encrypted using the BCP scheme, and this scheme is semantic secure under the DDH assumption over $\mathbb{Z}^*_{n^2}$ [4]. Hence, $\mathcal{A}$ cannot learn the stored data on the cloud platform. During the proxy re-encryption phase, the DSP and the CP each just performs a partial decryption and then uses some random value to blind the plaintext. Hence, even if $\mathcal{A}$ can collude with either DSP or CP, it cannot learn any information of the plaintext during this phase. Finally, the transformed ciphertexts are encrypted using an ABE scheme that is semantic secure under the l-BDHE assumption [14]. Hence, $\mathcal{A}$ cannot learn any information from the transformed ciphertext. Therefore, based on the DDH assumption over $\mathbb{Z}^*_{n^2}$ and the l-BDHE assumption, the proposed scheme satisfies the privacy property.

Theorem 3. *The proposed scheme satisfies the fine-grained access control property.*

Proof. After the proxy re-encryption, the computed result is blinded by two random values σ_1, σ_2, and these two values are encrypted by an ABE scheme. Hence, only the parties whose attribute satisfy the access structure can decrypt them and derive the computed result. This proves that fine-grained access control has been enfored on the computed results.

Theorem 4. *The proposed scheme satisfies the flexibility property.*

Proof. In theory, fine-grained access control also can be achieved through proxy re-encryption, i.e. the information can be re-encrypted so that only the designated recipients can decrypt it. However, this requires the re-encryption to be performed in real-time when the recipients' identities are known. In our proposed scheme, although proxy re-encryption is also used, it is only used to transform a ciphertext into an ABE ciphertext. In this way, the identities do not need to be known by the time these results are generated, and users can join afterwards. Hence, it is more versatile than the proxy re-encryption approach.

Theorem 5. *The proposed scheme satisfies the collusion resistance property.*

Proof. In our proposed scheme, when generating private keys for the DRs, the authority will assign a unique value r for each DR. Therefore, the private keys for different DRs will be associated with different values. Hence, they cannot put their attributes together to form a larger attribute sets. And this implies that multiple DRs cannot collude to gain more decryption privilege.

7 Efficiency Analyses

In this section, we analyze the performance of our scheme and compare it with two related works: called the PYS scheme [19] and the DTD scheme [21]. In the PYS scheme, a CP-ABE scheme is proposed based on the AND-Gate with wildcard access policy. In the DYD scheme, a privacy-preserving data processing scheme with flexible access control is proposed, using the KP-ABE scheme [10] for access control.

7.1 Communication Costs

As shown in Table 1, we compare the communication costs considering three aspects: size of public parameters, size of the secret key, and size of ciphertext. Let n be the number of attributes in the system. Let $|G_1|$ be the size of an element in the group G_1, and let $|G_2|$ be the size of an element in the group G_2. $|W|$ is the size of the access policy and λ is the number of attributes in W.

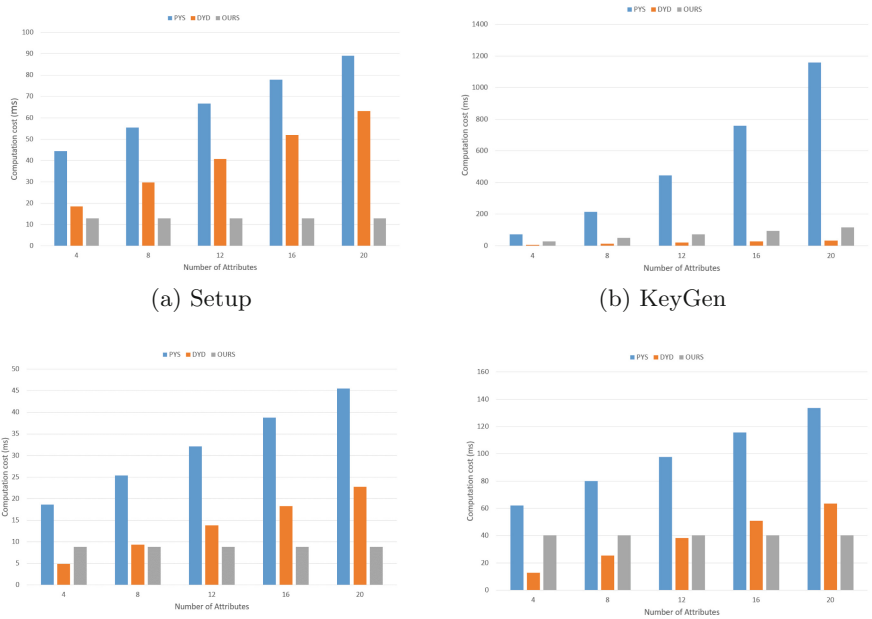
Although the size of public parameters and the size of secret key are larger in our proposed scheme. This information only generates once, and the size of ciphertext will dominates the communication costs. The size of ciphertext can remain constant in our proposed scheme, while it is proportional to the number of attributes in the DYD scheme. Hence, our scheme is more efficient in communication, compared with the DYD scheme. The PYS scheme just introduced an en efficient ABE scheme. Although it is more efficient than ours, it has not considered the privacy preserving data processing requirement and it cannot be used to solve the research problems in this paper.

Table 1. Comparison of communication costs

	Public parameters	Ciphertext	Secret Key
PYS Scheme	$(n+4)\|G_1\|+2\|G_2\|$	$3\|G_1\|+\|G_2\|+\|W\|$	$(n+6)\|G_1\|$
DYD Scheme	$(n+1)\|G_1\|+\|G_2\|$	$\lambda\|G_1\|+\|G_2\|+\|W\|$	$\lambda\|G_1\|$
Our scheme	$(3n+2)\|G_1\|+\|G_2\|$	$3\|G_1\|+\|G_2\|+\|W\|$	$(2n+2)\|G_1\|+\|G_2\|$

7.2 Computation Costs

To compare the computation costs, we have performed some experiments on the Windows platform with an Intel(R) Core(TM) i7-8550U CPU at 1.80 GHz and 8.00 GB RAM. Since the exponentiation operation and the bilinear pairing operation dominate the costs in computation, only these two operations are considered. In the experiment, we have used a Type A elliptic curve with 512 bits in the JPBC library [6]. The comparison of computation costs between these three schemes is shown in Fig. 2. In our proposed scheme, the computation costs are constant, while in the PYS and DYD schemes, they are linear to the system parameters, i.e. the number of attributes. Therefore, our proposed scheme is more practical, especially in large scale applications.



(a) Setup (b) KeyGen

(c) Encryption (d) Decryption

Fig. 2. Comparisons of Computation Cost

8 Conclusion

In this paper, we proposed a secure and efficient data processing scheme for cloud computing with fine-grained access control, using a homomorphic proxy re-encryption scheme and an efficient ABE scheme as the main building blocks. Our scheme supports privacy preserving computations on encrypted data, and fine-grained access control can be enforced on the computed results. Moreover, it is more efficient than the related schemes that satisfy similar properties. In particular, the size of ciphertexts and the decryption time for the computed results can be made constant. Therefore, our scheme contributes to a more practical data processing scheme for the cloud platform with fine-grained access control. For example, when it is used in smart grid, users' power consumption data can be stored and processed by the cloud platform and the results can only be utilized by the designated recipients, harmonizing security, scalability and usability.

Acknowledgement. We thank the anonymous reviewers for some helpful comments to improve the paper.

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP 2007), pp. 321–334. IEEE (2007)
2. Bogdanov, D., Kamm, L., Laur, S., Pruulmann-Vengerfeldt, P., Talviste, R., Willemson, J.: Privacy-preserving statistical data analysis on federated databases. In: Preneel, B., Ikonomou, D. (eds.) APF 2014. LNCS, vol. 8450, pp. 30–55. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06749-0_3
3. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_16
4. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40061-5_3
5. Chen, S., Wang, J., Yu, Z., Xu, H., Dong, C.: A cloud-assisted data processing scheme for smart grid with flexible access control. In: Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence, pp. 1–6 (2022)
6. De Caro, A., Iovino, V.: jPBC: Java pairing based cryptography. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 850–855. IEEE (2011)
7. Ding, W., Yan, Z., Deng, R.H.: Encrypted data processing with homomorphic re-encryption. Inf. Sci. **409**, 35–55 (2017)
8. Ge, C., Susilo, W., Liu, Z., Xia, J., Szalachowski, P., Fang, L.: Secure keyword search and data sharing mechanism for cloud computing. IEEE Trans. Dependable Secure Comput. **18**(6), 2787–2800 (2020)
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp. 169–178 (2009)

10. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
11. Hamza, R., Yan, Z., Muhammad, K., Bellavista, P., Titouna, F.: A privacy-preserving cryptosystem for IoT e-healthcare. Inf. Sci. **527**, 493–510 (2020)
12. Jagadeesh, K.A., Wu, D.J., Birgmeier, J.A., Boneh, D., Bejerano, G.: Deriving genomic diagnoses without revealing patient genomes. Science **357**(6352), 692–695 (2017)
13. Kamm, L., Willemson, J.: Secure floating point arithmetic and private satellite collision analysis. Int. J. Inf. Secur. **14**(6), 531–548 (2015)
14. Li, Q., Xia, B., Huang, H., Zhang, Y., Zhang, T.: TRAC: traceable and revocable access control scheme for mhealth in 5G-enabled IIoT. IEEE Trans. Industr. Inf. **18**(5), 3437–3448 (2021)
15. Lindell, Y.: Secure multiparty computation. Commun. ACM **64**(1), 86–96 (2020)
16. Liu, D., Yan, Z., Ding, W., Atiquzzaman, M.: A survey on secure data analytics in edge computing. IEEE Internet Things J. **6**(3), 4946–4967 (2019)
17. Lyu, L., Chau, S.C.-K., Wang, N., Zheng, Y.: Cloud-based privacy-preserving collaborative consumption for sharing economy. IEEE Trans. Cloud Comput. **10**(3), 1647–1660 (2020)
18. Nasiraee, H., Ashouri-Talouki, M.: Privacy-preserving distributed data access control for cloudiot. IEEE Trans. Dependable Secure Comput. **19**(4), 2476–2487 (2021)
19. Phuong, T.V.X., Yang, G., Susilo, W.: Hidden ciphertext policy attribute-based encryption under standard assumptions. IEEE Trans. Inf. Forensics Secur. **11**(1), 35–45 (2015)
20. Shen, J., Yang, H., Vijayakumar, P., Kumar, N.: A privacy-preserving and untraceable group data sharing scheme in cloud computing. IEEE Trans. Dependable Secure Comput. **19**(4), 2198–2210 (2021)
21. Ding, W., Yan, Z., Deng, R.H.: Privacy-preserving data processing with flexible access control. IEEE Trans. Dependable Secure Comput. **17**(2), 363–376 (2020)
22. Xia, Z., Yang, Q., Qiao, Z., Feng, F.: Quorum controlled homomorphic re-encryption for privacy preserving computations in the cloud. Inf. Sci. **621**, 58–73 (2023)
23. Zhang, W., Liu, S., Xia, Z.: A distributed privacy-preserving data aggregation scheme for smart grid with fine-grained access control. J. Inf. Secur. Appl. 103–118 (2022)