# A fully distributed hierarchical attribute-based encryption scheme

Ali Mohammad

Javad MOHAJERI

Ximeng LIU
*Singapore Management University*, xmliu@smu.edu.sg

Ximeng LIU

Published in Theoretical Computer Science, 2020 May, 815, 25-46.

https://doi.org/10.1016/j.tcs.2020.02.030

# A fully distributed hierarchical attribute-based encryption scheme

Mohammad Ali[a], Javad Mohajeri[b], Mohammad-RezaSadeghi[a], Ximeng Liu[cd]

[a] *Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran*
[b] *Electronics Research Institute, Sharif University of Technology, Tehran, Iran*
[c] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, PR China*
[d] *School of Information Systems, Singapore Management University, 178902, Singapore*

## Abstract

With the development of cloud computing, many enterprises have been interested in outsourcing their data to cloud servers to decrease IT costs and rise capabilities of provided services. To afford confidentiality and fine-grained data access control, attribute-based encryption (ABE) was proposed and used in several cloud storage systems. However, scalability and flexibility in key delegation and user revocation mechanisms are primary issues in ABE systems. In this paper, we introduce the concept of a fully distributed revocable ciphertext-policy hierarchical ABE (FDR-CP-HABE) and design the first FDR-CP-HABE scheme. Our scheme offers a high level of flexibility and scalability in the key delegation and user revocation phases. Moreover, our scheme is efficient and provides lightweight computation in the decryption phase. Indeed, by exploiting a computation outsourcing technique, most of the operations are executed by the powerful cloud server, and very few computations are left to the users. Also, the storage cost on the user side is significantly decreased as compared to similar schemes. Furthermore, using the hardness assumption of DBDH problem, we prove that our scheme is adaptively secure in the standard model. Our security analyses and implementation results indicate that our scheme is efficient, secure, and scalable.

## Keywords

Cloud computing, Hierarchical attribute-based encryption, Ciphertext-policy attribute-based encryption, Access control

## 1. Introduction

With the widespread applications of cloud computing technology, it is increasingly evolving to change the way how service is provided. The cloud computing technology benefits users in that it offers convenient access, storage and computation resources, distributed computing, increased operational efficiencies, etc. [1]. Therefore, to gain cost savings and the flexibility in investments on-demand, several enterprises benefit from cloud computing services to manage data, projects, contacts, etc. A 2016 RightScale report showed that 95 percent of surveyed enterprises are executing applications by using cloud computing. In principle, surveys indicate that more than 50 percent of all organizations consider the cloud computing to be a necessary part of their business models, and they are willing to dedicate half or more of their IT budget to this technology [2].

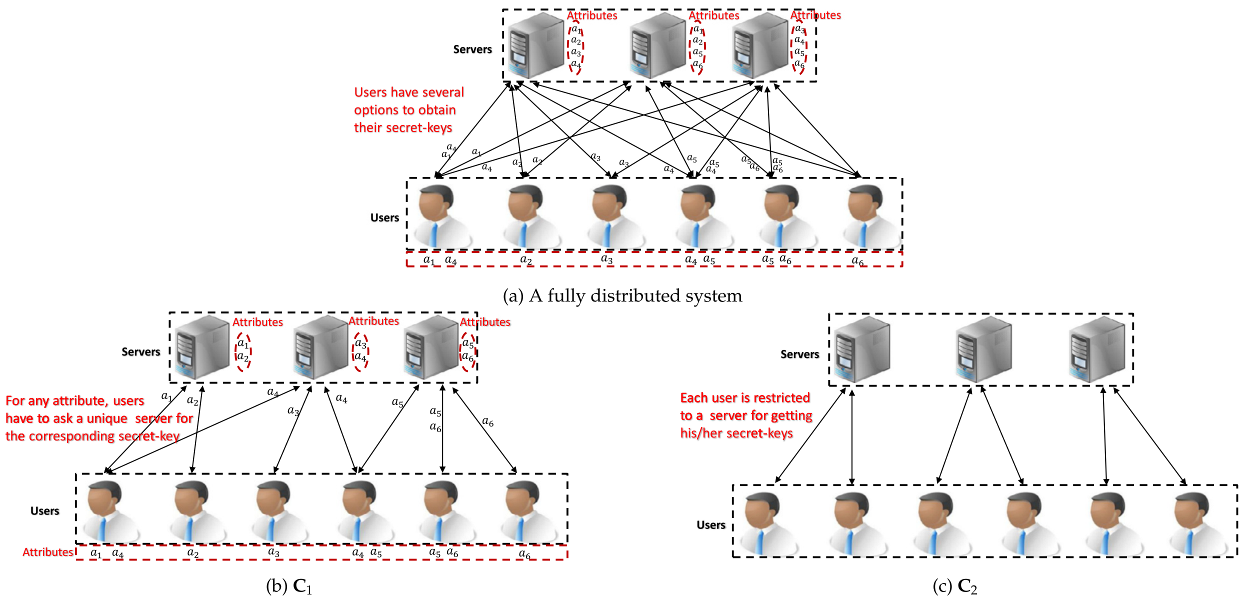

Servers
Attributes
Users have several options to obtain their secret-keys
Users
(a) A fully distributed system
For any attribute, users have to ask a unique server for the corresponding secret-key
(b) $\mathbf{C}_1$
Each user is restricted to a server for getting his/her secret-keys
(c) $\mathbf{C}_2$

**Fig. 1.** In parts (a) and (b), an arrow with attribute $a_i$ between a user and a server indicates that the server is able to provide the secret-key associated with attribute $a_i$ for the user. Also in part (c), each arrow between a user and a server means that the user can obtain his/her secret-keys from the server.

However, because a large amount of personal data are outsourced to cloud service providers, the concern about data privacy and confidentiality arises. To alleviate these issues, one obvious way is to encrypt data before uploading them to the cloud. Such methods also get in the way of controlling access rights over the outsourced encrypted data in that the cloud cannot be considered as a trusted entity, and it might attempt to analyze and access the individual data for some illegitimate purposes.

Attribute-based encryption (ABE) [3–5] is a one-to-many applicable cryptographic primitive that simultaneously attains data confidentiality and fine-grained access control. In an ABE scheme, an access control policy is defined to control the access rights of users. ABE schemes can be classified into two categories of key-policy ABE (KP-ABE) [6] and ciphertext-policy ABE (CP-ABE) [7]. In a KP-ABE scheme, secret-keys of a user is associated with an access control policy defined by the central authority, and ciphertexts are labeled with a set of descriptive attributes. A data user can decrypt a ciphertext only if its access control policy is satisfied by attributes of the ciphertext. While in a CP-ABE scheme, access control policies are defined by data owners and embedded in ciphertexts. Also, secret-keys of data users are associated with their attributes. A data user can decrypt a ciphertext only if the user's attribute set satisfies the access control policy associated with the ciphertext. It is clear that CP-ABE schemes are more convenient for both data owners and data users.

In a CP-ABE scheme, each user is specified by a set of descriptive attributes, and for each of the attributes, the user has to request the corresponding secret-key from a key generator authority. Also, in some data sharing applications in cloud computing environment, attributes of users are dynamic. This means that users may lose some of their attributes or obtain some new attributes at any time. So, traditional ABE systems with a single key generator authority seem not to be appropriate for large scale networks, and practical mechanisms for secret-key delegation and user revocation are required. Hierarchical ABE (HABE) system is a promising solution to address these problems. In a HABE system, there are several key generator servers that each of them are responsible for handling key delegation and user revocation associated with a specified set of attributes. To the best of the authors' knowledge, the existing ciphertext-policy HABE schemes can be divided into the following two categories:

- $\mathbf{C}_1$: For each attribute in the system, there is only one server to issue and revoke the secret-keys of users associated with the attribute [8–10] (see Fig. 1 part b).
- $\mathbf{C}_2$: For each user in the system, there is only one server to issue the users' secret-keys [11–13] (see Fig. 1 part c). Indeed the secret-keys gotten from different servers cannot be used together in the decryption phase.

However, both of the above categories have the following limitations that may cause serious problems:

- If because of some technical issues, or worse, physical catastrophes such as an earthquake or a fire, one of the serves cannot work for a temporary period of time, then in $\mathbf{C}_1$, the key delegation and user revocation services corresponding to the attributes associated with the server will be interrupted, and in $\mathbf{C}_2$ also the users associated with the server will be deprived of the services.

- In both of the categories, when the number of users requesting a server to generate their secret-keys is increased, because of the shortage of bandwidth and computational resources, the server may fail to provide the desired service.
- Assuming that, the physical presence of users is required for obtaining the secret-keys, we see that, in $\mathbf{C}_1$, a user may have to visit several servers which may be time-consuming, and in $\mathbf{C}_2$ also if a user is far from the specified server, then obtaining the secret-keys is difficult for the user.

In this paper, putting forward a fully distributed hierarchical CP-ABE system, we efficiently address the mentioned problems in the HABE systems. Our main contributions are listed as follows:

1. **Fully distributed key delegation and user revocation mechanisms**: Our proposed scheme addressees the mentioned issues in the existing HABE systems (see Fig. 1 part a). Our system offers the following advantages:
   - Each attribute in the system can be supported by several servers. This feature addresses the mentioned issues in $\mathbf{C}_1$. In fact, none of the attributes is restricted to a unique server, and each user has several options for obtaining his/her secret-key associated with an attribute.
   - Each user in the system can be serviced by several servers. This property resolves the mentioned issues in $\mathbf{C}_2$. Indeed, in our system, users have several options for getting their secret-keys, and secret-keys gotten from different servers can be used together in the decryption procedure.
2. **Scalable access control system**: Our scheme affords a high level of scalability. With increasing the number of users in a system, communication networks may encounter scalability challenges. In our proposed system, the central authority can generate some new domain authorities to improve the quality of services when more computational resources are required.
3. **Lightweight computation and storage overhead**: By using an outsourcing technique, we enable data users to outsource most of the decryption operations to the cloud service provider. Indeed, in the scheme, the cloud performs most of the operations without learning any sensitive information about the underlying data and users' secret-keys, and very few operations are left to data users. Also, in our scheme, the storage cost on the user side is very low as compared with similar schemes.
4. **Provable secure access control scheme**: By using standard hardness assumptions, we formally prove that our proposed scheme is adaptively secure in the standard model. We prove that all the polynomial-time adversaries cannot distinguish between two encrypted data unless they can solve the decisional bilinear Diffie-Hellman (DBDH) problem.

The rest of this paper is organized as follows: Section 2 reviews some related work. In Section 3, we introduce some required preliminaries. System model, threat model, system definition, design goals, and security definition are presented in Section 4. Section 5 presents our proposed fully distributed CP-HABE in detail. Also, the security and performance analyses are given in Section 6 and 7, respectively. In Section 8, we conclude this paper.

## 2. Related work

To achieve a convenient public-key encryption system, Shamir introduced the notion of identity-based encryption (IBE) scheme [14]. In an IBE scheme, public-key of a user is his/her identity (e.g., his/her e-mail address or phone number), and a sender can generate a ciphertext under the identity of a receiver without requesting the receiver's public-key beforehand. Boneh et al. proposed the first fully functional IBE scheme [15]. They provide a selectively secure IBE scheme in the random oracle model by exploiting the Weil pairing. Afterward, Waters proposed an adaptive secure IBE scheme in the standard model [16]. In addition to IBE schemes, several identity-based cryptographic primitives, such as proxy re-encryption [17,18], identity-based signature [19,20], and identity-based key agreement [21] have hitherto been designed.

The hierarchical identity-based encryption (HIBE) scheme [22,23] is another identity-based cryptographic primitive extending IBE schemes with key delegation to reduce key management burden on the key generator authority. In these schemes, each user is associated with an identity vector denoting its positions in the hierarchy. Users at a higher level are enabled to delegate secret-keys to their subordinates. Boneh et al. proposed a selectively secure HIBE scheme with constant-size ciphertext [24]. By adopting the methodology of dual system encryption proof, Waters [25] designed a fully secure HIBE scheme under simple assumptions. However, in identity-based schemes, there are some inherent issues. Firstly, identities of users may be revealed to the other parties that may threaten the privacy of users. Secondly, users normally have to prove their identities to the key authority by presenting some specified documents such as their passport, national card, etc. that may threaten their privacy. Thirdly, the resulted identity may not be unique if the associated identity information is not chosen properly. Fourthly, identities of users may be too long to remember, and keeping them in mind may be hard for users.

By replacing the identity in IBE with an attribute set, Sahi and Waters proposed the notion of attribute-based encryption (ABE) schemes [26]. In an ABE scheme, a data owner could share its data with several expected data users without knowing their identifiers and their public-keys. In fact, a data owner encrypts its data under a set of descriptive attributes and a threshold value $d$. Each user who has at least $d$ common attributes with the specified attribute set can recover the data. After the advent of ABE, two schemes [6,7] divided ABE schemes into two categories of key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE), respectively. Until then, all the existing ABE schemes were proven in the selective-model.

Lewko et al. [27] proposed the first adaptively secure ciphertext-policy and key-policy schemes. We refer the reader to [28–30] for more details on these cryptographic schemes.

In an ABE scheme, there is only one key generator authority to generate users' secret-keys. So, these schemes might not be desirable for large networks. To address this problem, Wang et al. proposed hierarchical ABE (HABE) schemes [8], by combining two notions ABE and HIBE. After introducing the HABE scheme, the idea has been applied in several ABE schemes. Liu et al. [9] by considering the concept of time into the combination of HABE and proxy re-encryption (PRE), proposed a time-based PRE scheme. The main benefit of the scheme is that data owners can be offline during the user revocation phase. In this scheme, each user can be identified by a set of attributes and a set of effective time periods determining how long the user is authorized for the attributes. Li et al. [10] proposed a multi-authority attribute-based data access control scheme. In their work, the decryption overhead on data users has been reduced significantly. Also, an efficient user revocation scheme has been proposed in the work. In schemes [8–10], there are several domain authorities that each of them administers a disjoint set of attributes. Data users have to request the secret-key corresponding to each of their attributes from the unique key generator authority. Also, when a user loses an attribute, the unique domain authority has to update the secret-keys of the unrevoked users corresponding to the attribute and generate an update-key for re-encrypting outsourced encrypted data associated with the attribute. By using an ABE scheme, a data owner can control data users' access rights under some attributes which can be organized logically as a single set. It causes some problems when a data owner does not want to allow that data users possessing some specific attributes together have access to its data. Bobba et al. addressed the problem by introducing attribute set-based encryption (ASBE) schemes [31]. Afterwards, Wan et al. [12] proposed a CP-HASBE by combining the notions of CP-ASBE and HABE. Huang et al. [11] proposed a data collaboration scheme, by using the hierarchical model in the key delegation mechanism. In the scheme, by using an outsourcing technique, decryption and signing computation overhead are reduced significantly.

It is notable that the term "hierarchical attribute-based encryption" has been used in two quite different categories. In the first category, like [8,10–12,32] and ours, to lighten the burden on key generator authority, several domain authorities with a hierarchy structure are considered. The main goal in designing these systems is to improve the scalability and flexibility of the system. However, in the second one [33–36], the attributes and files are assumed in a hierarchical relationship with each other, and there is just one key generator authority. In CP-HABE schemes [34,35], the shared data have a hierarchical structure. A group of files in these schemes is divided into some hierarchy subgroups, and the files in the same hierarchy structure can be encrypted under an integrated access policy. The main benefit of such schemes is saving encryption time and storage cost. We emphasize that the common term of HABE in these two categories is only a nominal similarity.

## 3. Preliminaries

In this section, we introduce some required preliminaries. The notations used in this section are described in Table 1.

### 3.1. Cryptographic backgrounds

**Bilinear map**: Suppose that $(G_1, +)$ and $(G_2, .)$ are two cyclic groups of a prime order $q$, and $P_0$ is a generator of $G_1$. A function $\hat{e} : G_1 \times G_1 \to G_2$ is called a bilinear map if it satisfies the following properties:

1. Non-degeneracy: $\hat{e}(P_0, P_0) \neq 1$.
2. Bilinearity: $\hat{e}(aP_1, bP_2) = \hat{e}(bP_1, aP_2) = \hat{e}(P_1, P_2)^{ab}$, for any $a, b \in \mathbb{Z}_q$ and $P_1, P_2 \in G_1$.
3. Computability: There is a polynomial time algorithm which computes $\hat{e}(P_1, P_2)$, for any $P_1, P_2 \in G_1$.

**Decisional bilinear Diffie-Hellman (DBDH) problem**: Let $(q, G_1, G_2, \hat{e})$ be as above, and let $P_0$ be a random generator of $G_1$. Consider three elements $aP_0$, $bP_0$ and $cP_0$ of $G_1$, where $a, b$ and $c$ are three uniform elements of $\mathbb{Z}_q$. The decisional bilinear Diffie-Hellman (**DBDH**) problem is to distinguish between $\hat{e}(P_0, P_0)^{abc}$ and $\hat{e}(P_0, P_0)^z$, where $z$ is a uniform element of $\mathbb{Z}_q$.

**Hardness assumption of DBDH problem**: Let $\mathcal{G}$ be a polynomial time algorithm that on input a security parameter $n$ outputs $(q, G_1, G_2, \hat{e})$, where $q$ is a prime number chosen according to $n$ and $G_1$, $G_2$, and $\hat{e}$ are the same as before. We say that the **DBDH** problem is hard relative to $\mathcal{G}$ if for any security parameter $n$, any probabilistic polynomial-time (PPT) distinguisher $\mathcal{D}$, and any $(q, G_1, G_2, \hat{e})$ generated by $\mathcal{G}(1^n)$, there is a negligible function *negl* such that:

$$\left|\Pr(\mathcal{D}(G_1, G_2, q, \hat{e}, aP_0, bP_0, cP_0, \hat{e}(P_0, P_0)^{abc}) = 1) - \Pr(\mathcal{D}(G_1, G_2, q, \hat{e}, aP_0, bP_0, cP_0, \hat{e}(P_0, P_0)^{z}) = 1)\right| \leq negl(n), \tag{1}$$

where $P_0$ is a random generator of $G_1$, and $a, b, c$ and $z$ are four uniform elements of $\mathbb{Z}_q$.

### 3.2. Access tree

Access trees are convenient to represent access policies [6]. In an access tree, leaf nodes are associated with a set of attributes, and any inner node represents a threshold value. Consider an access tree $\mathcal{T}$. Let $v_a$ be the leaf node associated

**Table 1**
Notations.

| Notation | Description |
|---|---|
| $\mathbb{U}$ | Universal attribute set |
| $\mathcal{T}$ | Access tree |
| $L_{\mathcal{T}}$ | Leaf node set of an access tree $\mathcal{T}$ |
| $R_{\mathcal{T}}$ | Root node of an access tree $\mathcal{T}$ |
| $\mathcal{T}_v$ | Subtree of $\mathcal{T}$ rooted at the node $v$ |
| $v_a$ | Leaf node associated with an attribute $a$ |
| $k_v$ | Threshold value of a node $v$ in an access tree |
| $ch_v$ | Children set of a node $v$ in an access tree |
| **CA** | The central authority |
| **CSP** | The cloud service provider |
| **DO** | Data owner |
| **DU** | Data user |
| **DA** | Domain authority |
| $\mathbf{DA}_i$ | The $i$-th domain authority |
| $sk_a$ | Master secret-key corresponding to an attribute $a$ |
| $PK_a$ | Public-key corresponding to an attribute $a$ |
| $MSK$ | Master secret-key of the **CA** |
| $PK$ | Public-parameter of the system |
| $(PK^{(i)}, MSK^{(i)})$ | Public-key and master secret-key of $\mathbf{DA}_i$ |
| $ID_i$ | Identifier of $\mathbf{DA}_i$ |
| $ID_{\mathbf{u}}$ | Identifier of a **DU** |
| $M, CT$ | A plaintext and a ciphertext, respectively |
| $SK_{i,a,\mathbf{u}}$ | A **DU**'s secret-key associated with an attribute $a$ and generated by $\mathbf{DA}_i$ |
| $\widetilde{SK}_{i,a,u}$ | An updated of the secret-key $SK_{i,a,\mathbf{u}}$ |
| $SK_{\mathbf{u}}$ | A **DU**'s secret-key set |
| $sk'_a, PK'_a$ | Updated master secret-key and public-key corresponding to an attribute $a$ |
| $\widetilde{CT}$ | A re-encrypted ciphertext |
| $TK_{CT}^{(\mathbf{u})}$ | Decryption token of a **DU** corresponding to the ciphertext $CT$ |
| $UK_a$ | Update-key associated with an attribute $a$ |
| $CT'$ | Partially decrypted ciphertext |

with an attribute $a$, $k_v$ be the threshold value of a node $v$, $ch_v$ is the children set of a node $v$, $R_{\mathcal{T}}$ be the root node of $\mathcal{T}$, $L_{\mathcal{T}}$ be the leaf node set of $\mathcal{T}$, and $\mathcal{T}_v$ be a subtree of $\mathcal{T}$ rooted at the node $v$.

Let $\mathbb{U}$ be the universal attribute set, and let $\mathcal{T}$ be an access tree. For a node $v$ in $\mathcal{T}$, consider a function $F_{\mathcal{T}_v} : 2^{\mathbb{U}} \to \{0, 1\}$ that for any attribute set $Att \in 2^{\mathbb{U}}$ the evaluation of $F_{\mathcal{T}_v}(Att)$ is performed as follows:

- When $v$ is the leaf node associated with an attribute $a$, $F_{\mathcal{T}_v}(Att) = 1$ if and only if $a \in Att$.
- When $v$ is a non-leaf node of $\mathcal{T}$ with threshold value $k_v$, $F_{\mathcal{T}_v}(Att) = 1$ if and only if $v$ has at least $k_v$ children $c_1, \ldots, c_{k_v}$ that $F_{\mathcal{T}_{c_i}}(Att) = 1$, for any $i = 1, \ldots k_v$.

We say that an access tree $\mathcal{T}$ is satisfied by an attribute set $Att$ and denote it by $F_{\mathcal{T}}(Att) = 1$ if $F_{\mathcal{T}_{R_{\mathcal{T}}}}(Att) = 1$.

Given an access tree $\mathcal{T}$ and a prime number $q$, we denote the algorithm for distributing a secret $r$ according to $\mathcal{T}$ and $q$ as:

$$\mathbf{Share}(r, q, \mathcal{T}) \to \{q_{v_a}(0)\}_{v_a \in L_{\mathcal{T}}}. \tag{2}$$

The algorithm assigns a polynomial to each node $v$ in $\mathcal{T}$ with respect to $k_v$ in a top-down style as follows:

- If $v = R_{\mathcal{T}}$, then $q_v$ is a $(k_v - 1)$-degree polynomial that $q_v(0) = r$, and its remaining coefficients are chosen randomly from $\mathbb{Z}_q$.
- If $v$ is the $i$-th child of a node $v'$, then $q_v$ is a $(k_v - 1)$-degree polynomial that $q_v(0) = q_{v'}(i)$, and its remaining coefficients are chosen randomly from $\mathbb{Z}_q$.

When $\mathbf{Share}(r, q, \mathcal{T})$ stops, a value $q_{v_a}(0)$ is assigned to the leaf node $v_a$, for any $v_a \in L_{\mathcal{T}}$.

Given an access tree $\mathcal{T}$, an attribute set $Att$, a tuple $(n, q, G_1, G_2, \hat{e})$ of bilinear map as in Subsection 3.1, and a set of values $\{\hat{e}(P_1, P_2)^{q_{v_a}(0)}\}_{a \in Att}$, where $\{q_{v_a}(0)\}_{v_a \in L_{\mathcal{T}}}$ is an output of $\mathbf{Share}(r, q, \mathcal{T})$ and $P_1, P_2 \in G_1$ are two arbitrary elements. We denote the algorithm for calculating $\hat{e}(P_1, P_2)^r$ as

$$\mathbf{Combine}(q, \mathcal{T}, \{\hat{e}(P_1, P_2)^{q_{v_a}(0)}\}_{a \in Att}) \to \hat{e}(P_1, P_2)^r. \tag{3}$$

This algorithm performs the following steps in a bottom-top style according to $\mathcal{T}$ as follows:

- If $Att$ does not satisfy $\mathcal{T}$, the algorithm aborts.

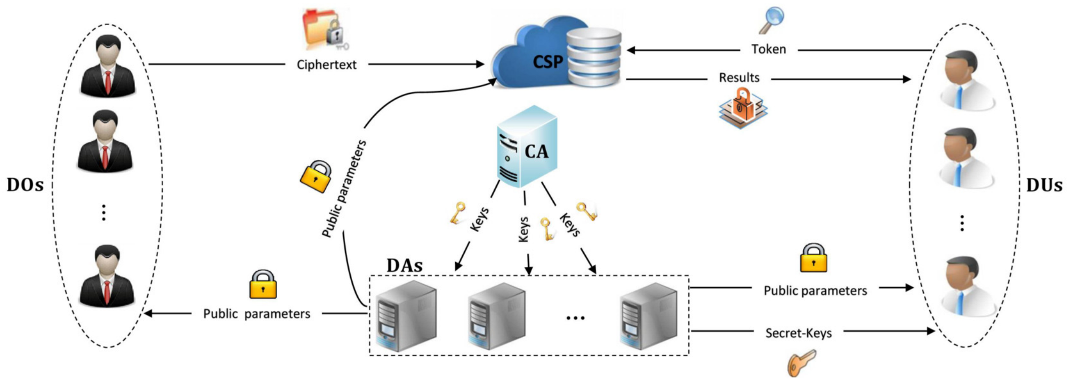


**Fig. 2.** Basic FDR-CP-HABE system model.

- Otherwise, it executes below:
  - Assign $\hat{e}(P_1, P_2)^{q_{v_a}(0)}$ to $v_a$ for any $a \in Att$.
  - For any inner node $v$ with children set $ch_v = \{v_{i_1}, \ldots, v_{i_{|ch_v|}}\}$, check whether $F_{\mathcal{T}_v}(Att) = 1$ or not. If so, compute

$$\prod_{j=1}^{|ch_v|} (\hat{e}(P_1, P_2)^{q_{v_{i_j}}(0)})^{l_{i_j}} = \hat{e}(P_1, P_2)^{\sum_{j=1}^{|ch_v|} l_{i_j} q_{v_{i_j}}(0)} = \hat{e}(P_1, P_2)^{q_v(0)}, \tag{4}$$

  and assign it to $v$, where

$$l_{i_j} = \prod_{\substack{t=1 \\ i_t \neq i_j}}^{k_v} \frac{-i_j}{i_t - i_j}. \tag{5}$$

The **Combine** algorithm assigns the value $\hat{e}(P_1, P_2)^r$ to the root node of $\mathcal{T}$.

## 4. Problem formulation

In this section, we first define a revocable CP-HABE systems and then introduce the concept of a *fully distributed revocable* CP-HABE (FDR-CP-HABE) system for the first time. Then, we present the system model, threat model, system definition, security model, design goals, and the security definition of an FDR-CP-HABE scheme. Table 1 presents the notations used in this section and our proposed construction in Section 5.

### *4.1. Revocable CP-HABE systems*

A revocable CP-HABE system comprises five types of entities, namely: A Central Authority (**CA**), multiple Domain Authorities (**DA**s), a Cloud Service Provider (**CSP**), several Data Owners (**DO**s) and Data Users (**DU**s) (see Fig. 2). The tasks for each of the entities are described as follows:

- **CA**: It initializes public parameters of and master secret-keys for **DA**s of the system.
- **DAs**: They generate secret-keys for **DU**s according to their attributes and revoke their access privileges when **DU**s lose some of their attributes.
- **CSP**: It provides storage, computation, and fine-grained access control services for **DO**s and **DU**s.
- **DO**s: Any **DO** defines an access control policy, encrypts its data under the defined access policy, and outsources the encrypted data to the **CSP**.
- **DU**s: Each **DU** is specified by a set of descriptive attributes. They obtain their secret-keys from **DA**s with respect to their attributes. They can decrypt a ciphertext if and only if the attribute set associated with their secret-keys satisfies the access control policy associated with the ciphertext.

A CP-HABE system consists of the following five phases [8] (see Fig. 3):

1. **System Setup (Phase 1)**: The **CA** manages this phase. It initializes public parameters of the system and generates master secret-keys of **DA**s.
2. **Key delegation (Phase 2)**: In this phase, a **DU** joins to the system and selects a unique identifier. Then, according to its attributes, it obtains some secret-keys from **DA**s.

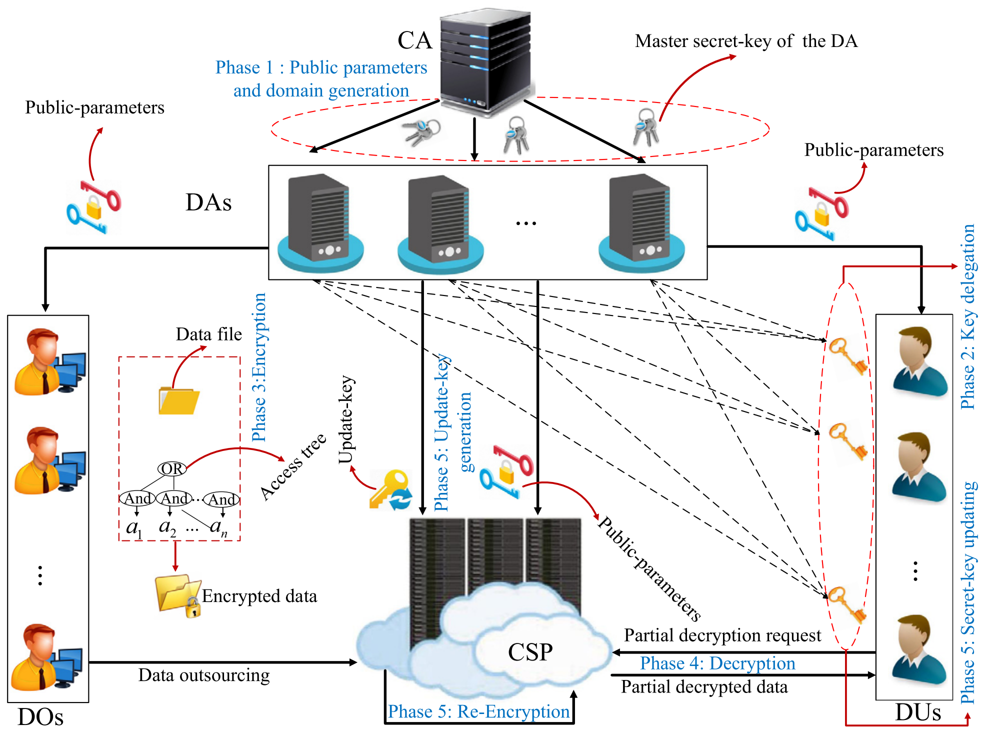


**Fig. 3.** Architecture of our FDR-CP-HABE scheme.

3. **Encryption (Phase 3)**: **DO**s execute this phase. They first define an access control policy and then encrypt their data under it. Finally, they outsource their encrypted data to the **CSP**.
4. **Decryption (Phase 4)**: In this phase, **DU**s decrypt the encrypted data in the **CSP** by using the secret-keys obtained in Phase 3.
5. **User revocation (Phase 5)**: This phase can be divided into three parts:
   - **Part 1**: When a **DU** loses an attribute, **DA**s update parameters of the system associated with the attribute. Then, **DA**s generate an update-key and send it to the **CSP**.
   - **Part 2**: **DA**s update secret-keys of authorized **DU**s associated with the attribute.
   - **Part 3**: The **CSP** re-encrypts the outsourced ciphertexts associated with the attribute by using the update-key generated in Part 1.

## *4.2. Fully distributed revocable CP-HABE (FDR-CP-HABE) system*

Consider the revocable CP-HABE system introduced in Section 4.1. We say that the system is *fully distributed* if the following conditions hold:

- For any attribute $a$ in the system and any **DU** possessing the attribute, the secret-key of the **DU** associated with the attribute can be generated by several **DA**s.
- The attribute secret-keys generated by different **DA**s can be used together in **Decryption** phase.
- Part 1 and 2 of **Revocation** phase can be executed by several **DA**s supporting the attribute.

It is clear that an FDR-CP-HABE system provides ideal flexibility in key delegation and user revocation phases (Phases 2 and 5). To the best of our knowledge, none of the existing HABE schemes support this feature. Indeed, as we mentioned in Section 1, in the current schemes, either each attribute is managed by a unique **DA**, or each **DU** has to obtain its secret-keys from a unique **DA**. In Section 5, we present the first FDR-CP-HABE scheme in detail.

## *4.3. System model*

The architecture of our proposed FDR-CP-HABE scheme is described in Fig. 3. Similar to Subsection 4.1, in our proposed scheme, there are five types of entities **CA**, **DA**s, **CSP**, **DO**s, and **DU**s. Also, it consists of five phases **System setup**, **Key delegation**, **Encryption**, **Decryption**, and **User revocation**.

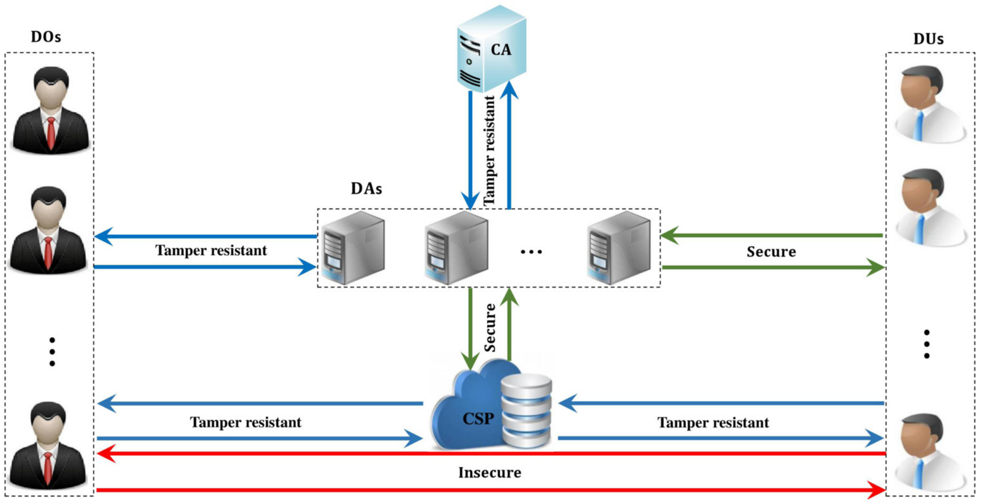


**Fig. 4.** Communication channels in our proposed schemes.

As we have shown in Fig. 3, in Phase 1, the **CA** generates some new **DA**s and provide them with some master secret-keys and public parameters of the system. Then the **DA**s spread the public parameters to the other entities through some communication channels. For each of its attributes, a **DU** should ask some **DA**s supporting the attribute to generate the corresponding attribute secret-key, in Phase 2. In this phase, the DU can make query to each **DA** that has the attribute, and secret-keys obtained from different **DA**s can be used together in the decryption phase. In Phase 3, a **DO** defines an access tree and encrypts its data under the access tree. The encrypted data is outsourced to the **CSP**. In Phase 4, a **DU** whose attributes satisfy the access tree associated with a ciphertext can ask the **CSP** to decrypt the ciphertext partially. The **CSP** performs most of the operations in **Decryption** phase without learning any unauthorized information about the underlying data and the **DU**'s secret-keys, and very few operations are left to the **DU**. Using the partially decrypted ciphertext, the **DU** recovers the associated message. In Phase 5, when a **DU** loses an attribute, its access right is revoked in three parts. Part 1: A **DA** first changes the parameters of the system associated with the attribute and generates an update-key corresponding to the attribute. The update-key is given to the **CSP**. In Part 2: according to the new parameters, **DA**s update secret-keys of **DU**s who still have the attribute. In Part 3: the **CSP** re-encrypts data which their access tree has a leaf node corresponding to the revoked attribute. Note that, any **DA** can handle first and second parts of this phase, independently. In fact, any **DA** can update secret-keys of any **DU** even if the secret-keys is not issued by the **DA**.

## 4.4. *Threat model*

The **CA** and **DA**s are assumed to be trusted. They never collude with the **CSP** and **DU**s. The **CSP** is assumed to be semi-trusted. It always executes the given protocols correctly, and it does not collude with **DU**s [37–41]. However, it sometimes tries to find further information about the stored data. All **DU**s are assumed to be malicious. They may collude with each other to obtain unauthorized access to the outsourced data.

As we have shown in Fig. 4, the communication channels between **DA**s and the **CSP** and also the channels between **DA**s and **DU**s are assumed to be secure. The transmitted data through the channels neither can be changed nor eavesdropped by some malicious parties. The channels between **DA**s and the **CA**, **DA**s and **DO**s, the **CSP** and **DO**s, and the **CSP** and **DU**s are assumed to be tamper-resistant. That means, transmitted data through the channels may be eavesdropped by some adversaries, but it is not definitely tampered with. Also, we assume that the communication channels between **DO**s and **DU**s are fully insecure.

## 4.5. *Definition of our proposed FDR-CP-HABE system*

Our proposed FDR-CP-HABE scheme consists of the following ten algorithms shown in Fig. 5. Based on the presented system model, in the following, we define our proposed scheme.

1. **Setup**$(1^n, \mathbb{U}) \rightarrow (PK, MSK)$: The **CA** operates this algorithm. It takes the security parameter $n$ and the universal attribute set $\mathbb{U}$ as inputs. It generates the global public parameters $PK$ and the master secret-key $MSK$.
2. **CreateDA**$(PK, MSK, \{ID_i\}_{i=1}^k, \{\mathbb{A}_i\}_{i=1}^k) \rightarrow \{PK^{(i)}, MSK^{(i)}\}_{i=1}^k$: To establish $k \in \mathbb{Z}$ new domain authorities **DA**$_1, \ldots,$ **DA**$_k$, the **CA** executes this algorithm. It takes public parameters $PK$, the master secret-key $MSK$, identifiers $\{ID_i\}_{i=1}^k$, and attribute sets $\{\mathbb{A}_i\}_{i=1}^k$, as inputs, where $ID_i$ and $\mathbb{A}_i$ are the identifier and the attribute set of the $i$-th new **DA**. It outputs a public-key $PK^{(i)}$ and a master secret-key $MSK^{(i)}$ for **DA**$_i$, $i = 1, \ldots, k$.

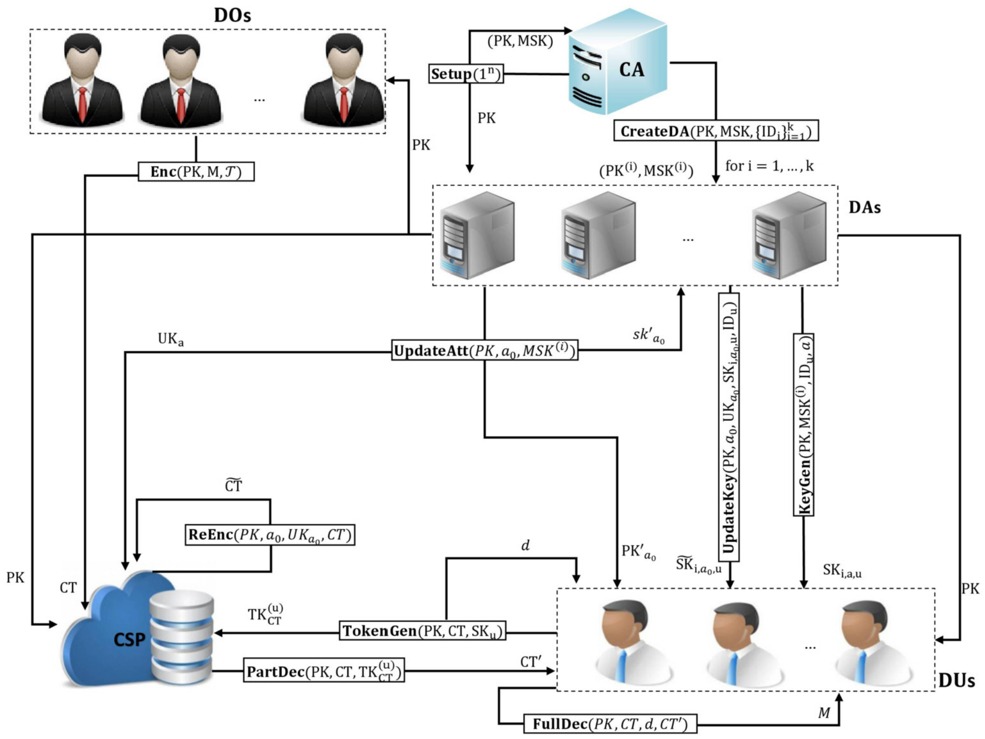


**Fig. 5.** Workflow of our proposed scheme.

3. **KeyGen**$(PK, MSK^{(i)}, ID_{\mathbf{u}}, a) \rightarrow SK_{i,a,\mathbf{u}}$: This algorithm can be operated by a domain **DA**$_i$. It takes public parameters of the system, $PK$, master secret-key of **DA**$_i$, $MSK^{(i)}$, identifier of the **DU**, $ID_{\mathbf{u}}$, and an attribute $a$ as inputs. It outputs a secret-key $SK_{i,a,\mathbf{u}}$ if $a \in \mathbb{A}_i$, where $\mathbb{A}_i$ is the attribute set of **DA**$_i$. Otherwise, it outputs $\perp$.
4. **Enc**$(PK, M, \mathcal{T}) \rightarrow CT$: A **DO** runs this algorithm. It takes public parameters of the system, $PK$, a message $M$, and an access tree $\mathcal{T}$ as inputs. The algorithm outputs a ciphertext $CT$ corresponding to the message and the access tree.
5. **TokenGen**$(PK, CT, SK_{\mathbf{u}}) \rightarrow (d, TK_{CT}^{(\mathbf{u})})$: To generate a decryption token for the **CSP**, a **DU** executes this algorithm. The inputs of the algorithm are system public parameters $PK$, a ciphertext $CT$, and secret-key set of the **DU**, $SK_{\mathbf{u}}$. If the attribute set associated with $SK_{\mathbf{u}}$ satisfies the access tree of $CT$, then the algorithm returns a decryption token $TK_{CT}^{(\mathbf{u})}$ and its associated private-key $d$. Otherwise, it outputs an error message $\perp$.
6. **PartDec**$(PK, CT, TK_{CT}^{(\mathbf{u})}) \rightarrow CT'$: The **CSP** operates this algorithm. It takes system public parameters $PK$, a ciphertext $CT$, and a decryption token $TK_{CT}^{(\mathbf{u})}$ as inputs. It outputs a partially decrypted ciphertext $CT'$ or an error message $\perp$.
7. **FullDec**$(PK, CT, d, CT') \rightarrow M$: After obtaining a partially decrypted ciphertext form the **CSP**, a **DU** can execute this algorithm. It takes system public parameters $PK$, a ciphertext $CT$, a private-key $d$, and a partially decrypted ciphertext $CT'$. It outputs the message corresponding to $CT$.
8. **UpdateAtt**$(PK, a_0, MSK^{(i)}) \rightarrow (UK_{a_0}, sk'_{a_0}, PK'_{a_0})$: This algorithm is executed by a domain authority **DA**$_i$. The inputs of the algorithm are system public parameters $PK$, an attribute $a_0$ revoked from a **DU**, and the master secret-key of **DA**$_i$, $MSK^{(i)}$. It updates parameters of the system associated with the attribute and generates an update-key $UK_{a_0}$.
9. **UpdateKey**$(PK, a_0, UK_{a_0}, SK_{i,a_0,\mathbf{u}}, ID_{\mathbf{u}}) \rightarrow \widetilde{SK}_{i,a,\mathbf{u}}$: This algorithm is executed by **DA**s. It takes system public parameters, $PK$, an attribute $a_0$, an update-key $UK_{a_0}$, a secret-key of a **DU** associated with the attribute, $SK_{i,a_0,\mathbf{u}}$, and the **DU**'s identifier, $ID_{\mathbf{u}}$. It returns an updated secret-key $\widetilde{SK}_{i,a_0,\mathbf{u}}$.
10. **ReEnc**$(PK, a_0, UK_{a_0}, CT) \rightarrow \widetilde{CT}$: The **CSP** runs this algorithm. The inputs of the algorithm are system public parameters $PK$, an attribute $a_0$ revoked from a **DU**, an update key $UK_{a_0}$, and a ciphertext $CT$ that its access tree has a leaf-node associated with $a_0$. The output of the algorithm is a re-encrypted ciphertext $\widetilde{CT}$.

### *4.6. Design goals*

In the following, we list our main design goals:

- **Fine-grained data access control**: In practice, several **DO**s outsource their sensitive data to the **CSP**, and several **DU**s are interested in accessing the data. **DO**s should be able to determine the authorized **DU**s to access their individual data by defining an access control policy.

- **Data confidentiality**: Our proposed scheme must be semantically secure. In other words, unauthorized **DU**s and the **CSP** can not learn any partial information about the plaintexts from the corresponding ciphertexts.
- **Collusion resistance**: A group of unauthorized **DU**s may collude to decrypt some outsourced encrypted data. Our scheme must be resistant against such collusion attacks.
- **Scalability and flexibility**: Our scheme should provide a high level of flexibility and scalability in **Key delegation** and **User revocation** phases. To achieve this goal, our scheme should be fully distributed as we mentioned in Section 4.2.
- **High performance**: The computational and storage cost on **DU**s should be low enough such that they can easily use the system.

## 4.7. Security definition of an FDR-CP-HABE scheme

We define the semantic security of an FDR-CP-HABE scheme in terms of the following experiment denoted by $\mathbf{HABE}^{cpa}_{\mathcal{A},\Pi}(n)$, where $\Pi$ is an FDR-CP-HABE scheme, $\mathcal{A}$ is a PPT adversary, and $n$ is a security parameter of the system. The experiment consists of the following five steps:
**The CPA indistinguishability experiment $\mathbf{HABE}^{cpa}_{\mathcal{A},\Pi}(n)$:**

1. **Setup:** A challenger runs **Setup**$(1^n)$ and **CreateDA**$(PK, MSK, \{ID_i\}_{i=1}^k)$ algorithms and gives the public parameters of the system, $PK$, and identifiers and public-keys of the generated **DA**s to $\mathcal{A}$.
2. **Phase 1:** The adversary $\mathcal{A}$ adaptively makes a polynomial number of queries to the following oracle. The challenger keeps a list $L_{ID_{\mathbf{u}}}$, for any **DU** with identifier $ID_{\mathbf{u}}$, and two black lists $L_{B_1}$ and $L_{B_2}$. All the lists are initially empty:
$\mathcal{O}_{\mathbf{KeyGen}}(ID_i, ID_{\mathbf{u}}, a)$: The challenger first checks whether $ID_u \in L_{B_1}$ or not. If so, it aborts. Otherwise, the challenger runs **KeyGen**$(PK, MSK^{(i)}, ID_{\mathbf{u}}, a)$ and gives the queried secret-key to $\mathcal{A}$. It also adds attribute $a$ into $L_{ID_{\mathbf{u}}}$ and identity $ID_u$ into $L_{B_2}$.
3. **Challenge:** Adversary $\mathcal{A}$ declares an access tree $\mathcal{T}$ and two equal length messages $M_0$ and $M_1$. The challenger checks whether there is a list $L_{ID_{\mathbf{u}}}$ satisfying $\mathcal{T}$ or not. If so, the challenger aborts. Otherwise, it flips a random coin $b \in \{0, 1\}$ and encrypts $M_b$ under $\mathcal{T}$. The generated ciphertext $CT_b$ is returned to $\mathcal{A}$.
4. **Phase 2:** $\mathcal{A}$ submits more queries to the following oracles, for polynomially many times:
   - $\mathcal{O}_{\mathbf{KeyGen}}(ID_i, ID_{\mathbf{u}}, a)$: The challenger checks if $L_{ID_{\mathbf{u}}} \cup \{a\}$ satisfies $\mathcal{T}$ or not. If not, it answers the query the same as in phase 1. Otherwise, it aborts.
   - $\mathcal{O}_{\mathbf{TokenGen}}(CT_b, ID_{\mathbf{u}}, Att)$: The challenger first checks whether $ID_u \in L_{B_2}$ or not. If so, it aborts. Otherwise, it generates $SK_{\mathbf{u}} = \{SK_{i,a,u}\}_{a\in Att}$ by using **KeyGen** algorithm. Then, it executes **TokenGen**$(PK, CT, SK_{\mathbf{u}}) \to (d, TK^{(\mathbf{u})}_{CT_b})$ and gives $TK^{(\mathbf{u})}_{CT_b}$ to the adversary. Finally, it adds $ID_{\mathbf{u}}$ into $L_{B_1}$.
5. **Guess:** The adversary outputs a guess $b' \in \{0, 1\}$ of $b$.

The output of the experiment is defined to be 1 if $b = b'$, and 0 otherwise. In the former case, we say the adversary $\mathcal{A}$ succeeds and denote it by $\mathbf{HABE}^{cpa}_{\mathcal{A},\Pi}(n) = 1$.

**Definition 1.** An FDR-CP-HABE scheme $\Pi$ is called CPA-secure if for all PPT adversaries $\mathcal{A}$ there is a negligible function *negl* such that:

$$\Pr(\mathbf{HABE}^{cpa}_{\mathcal{A},\Pi}(n) = 1) \le \frac{1}{2} + negl(n), \tag{6}$$

where $n$ is the security parameter of the scheme, and the probability is taken over the randomness of the experiment, as well as the randomness used by the adversary $\mathcal{A}$.

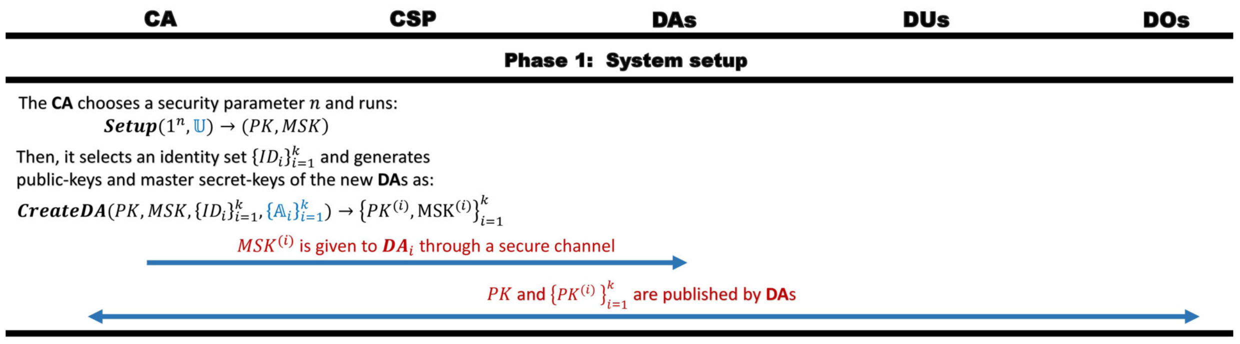
## 5. Our construction

In this section, we present our proposed FDR-CP-HABE scheme in detail.

### 5.1. System setup

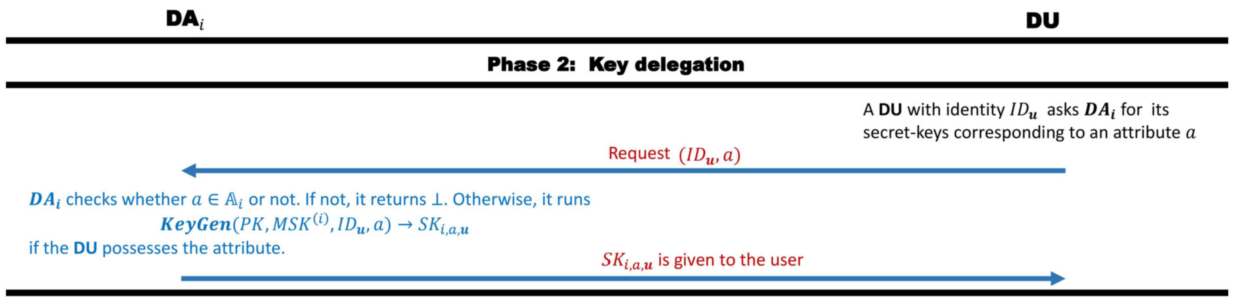
This phase is executed by the **CA**. It first selects security parameter $n$ and a universal attribute set $\mathbb{U}$. Then, it runs **Setup**$(1^n, \mathbb{U})$ algorithm to generated public parameters and the master secret-key of the system. Then, it generates $k \in \mathbb{Z}$ new domain authorities, $\mathbf{DA}_1, \ldots, \mathbf{DA}_k$. It chooses a unique identifier $ID_i \in G_1$ and an attribute set $\mathbb{A}_i \subseteq \mathbb{U}$ for $\mathbf{DA}_i$, $i = 1, \ldots, k$, and provides the public-key and the master secret-key of $\mathbf{DA}_i$ by running **CreateDA**$(PK, MSK, \{ID_i\}_{i=1}^k, \{\mathbb{A}_i\}_{i=1}^k)$ (see Fig. 6). The mentioned two algorithms are defined as follows.

**Setup**$(1^n, \mathbb{U})$: This algorithm runs $(q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n)$ and then selects $x, sk_0 \in \mathbb{Z}_q$ and $X, P_0, P_1, P_2 \in G_1$ uniformly at random and chooses a universal attribute set $\mathbb{U}$. Afterwards, for any attribute $a \in \mathbb{U}$, it selects a random element $sk_a \in \mathbb{Z}_q$ and calculates $PK_a = sk_a P_0$. Finally, it outputs



CA
CSP
DAs
DUs
DOs
Phase 1: System setup
The CA chooses a security parameter $n$ and runs:
$\boldsymbol{Setup}(1^n, \mathbb{U}) \rightarrow (PK, MSK)$
Then, it selects an identity set $\{ID_i\}_{i=1}^k$ and generates public-keys and master secret-keys of the new DAs as:
$\boldsymbol{CreateDA}(PK, MSK, \{ID_i\}_{i=1}^k, \{\mathbb{A}_i\}_{i=1}^k) \rightarrow \{PK^{(i)}, MSK^{(i)}\}_{i=1}^k$
$MSK^{(i)}$ is given to $\boldsymbol{DA_i}$ through a secure channel
$PK$ and $\{PK^{(i)}\}_{i=1}^k$ are published by DAs

**Fig. 6.** Procedures of Phase 1 of the scheme.



DA$_i$
DU
Phase 2: Key delegation
A DU with identity $ID_{\boldsymbol{u}}$ asks $\boldsymbol{DA_i}$ for its secret-keys corresponding to an attribute $a$
Request $(ID_{\boldsymbol{u}}, a)$
$\boldsymbol{DA_i}$ checks whether $a \in \mathbb{A}_i$ or not. If not, it returns $\perp$. Otherwise, it runs
$\boldsymbol{KeyGen}(PK, MSK^{(i)}, ID_{\boldsymbol{u}}, a) \rightarrow SK_{i,a,\boldsymbol{u}}$
if the DU possesses the attribute.
$SK_{i,a,\boldsymbol{u}}$ is given to the user

**Fig. 7.** Procedures of Phase 2 of the scheme.

$$PK = \left(q, G_1, G_2, \hat{e}, n, P_0, P_1, P_2, Q_0, Q_1, g_0, g_1, \{PK_a\}_{a\in\mathbb{U}}\right) \tag{7}$$

and

$$MSK = (X, sk_0, x, SK_1, SK_2, \{sk_a\}_{a\in\mathbb{U}}) \tag{8}$$

as public parameters and master secret-key of the system, where $Q_0 = sk_0P_0$, $Q_1 = xP_0$, $g_0 = \hat{e}(Q_0, P_1)$, $g_1 = \hat{e}(Q_1, -Q_0)$, $SK_1 = sk_0P_1$, and $SK_2 = xQ_0$.

**CreateDA**$(PK, MSK, \{ID_i\}_{i=1}^k, \{\mathbb{A}_i\}_{i=1}^k)$: This algorithm selects $k$ uniform elements $sk_1, \dots, sk_k \in \mathbb{Z}_q$ and computes

$$SK^{(i)} = SK_1 + X + sk_iID_i \tag{9}$$

and

$$PK^{(i)} = SK_2 + X + sk_iID_i. \tag{10}$$

It returns $\{PK^{(i)}, MSK^{(i)}\}_{i=1}^k$, where

$$MSK^{(i)} = \left(SK^{(i)}, \{sk_a\}_{a\in\mathbb{A}_i},\right) \tag{11}$$

is the master secret-key of the **DA**$_i$, $i = 1, \dots, k$.

**Remark 2.** Our proposed system offers a high level of scalability in the key delegation and user revocation phases. This means, it has the property to handle a growing number of queries by adding its resources. Indeed, when the number of **DU**s requesting to a **DA** providing secret-keys associated with attribute set $\mathbb{A}$ is increased, to improve the computational resources of the system, the **CA** can establish some new **DA**s supporting the attribute set $\mathbb{A}$. Therefore, by using this technique, the computational and communication burden can be distributed among the new **DA**s.

## *5.2. Key delegation*

When a **DU** joins to the system, it should select a unique identifier $ID_{\mathbf{u}} \in G_1$. Then, for each of its attributes, it should request the corresponding secret-key from a domain **DA**$_i$ supporting the attribute. **DA**$_i$ checks whether the **DU** has the attribute or not. If not, it aborts. Otherwise, it runs **KeyGen**$(PK, MSK^{(i)}, ID_{\mathbf{u}}, a)$ algorithm as follows and generates the requested secret-key (see Fig. 7).

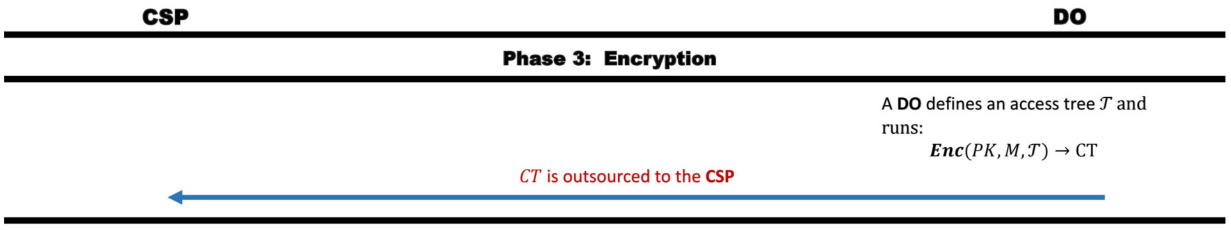

CSP
DO
Phase 3: Encryption
A DO defines an access tree $\mathcal{T}$ and runs:
$\boldsymbol{Enc}(PK, M, \mathcal{T}) \rightarrow \text{CT}$
$CT$ is outsourced to the CSP

**Fig. 8.** Procedures of Phase 3 of the scheme.

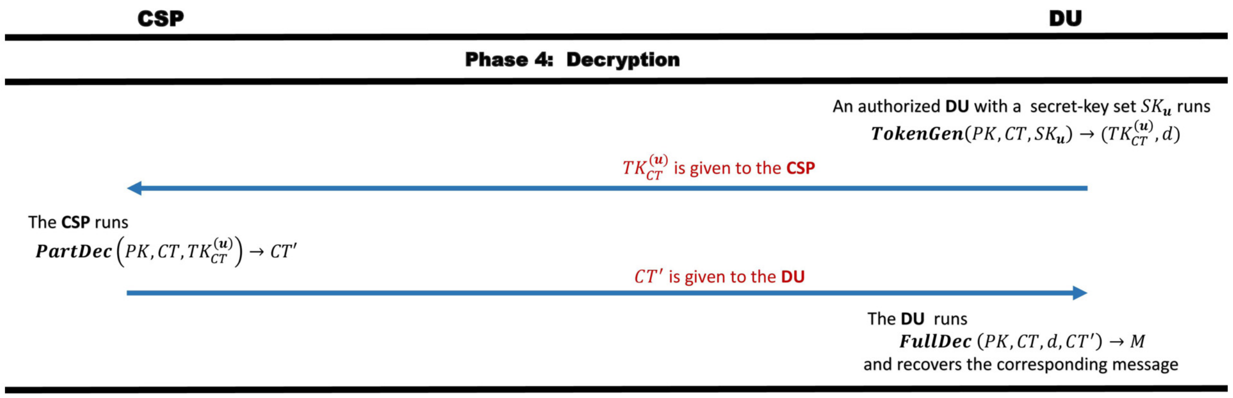

CSP
DU
Phase 4: Decryption
An authorized DU with a secret-key set $SK_{\boldsymbol{u}}$ runs
$\boldsymbol{TokenGen}(PK, CT, SK_{\boldsymbol{u}}) \rightarrow (TK_{CT}^{(\boldsymbol{u})}, d)$
$TK_{CT}^{(\boldsymbol{u})}$ is given to the CSP
The CSP runs
$\boldsymbol{PartDec}\left(PK, CT, TK_{CT}^{(\boldsymbol{u})}\right) \rightarrow CT'$
$CT'$ is given to the DU
The DU runs
$\boldsymbol{FullDec}\,(PK, CT, d, CT') \rightarrow M$
and recovers the corresponding message

**Fig. 9.** Procedures of Phase 4 of the scheme.

**KeyGen**$(PK, MSK^{(i)}, ID_{\mathbf{u}}, a)$: This algorithm at first checks whether $a \in \mathbb{A}_i$ or not. If not, it returns $\perp$. Otherwise, it outputs a secret-key corresponding to the tuple $(a, ID_{\mathbf{u}}, \mathbf{DA}_i)$ as follows:

$$SK_{i,a,\mathbf{u}} = SK^{(i)} + sk_a ID_{\mathbf{u}}. \tag{12}$$

## 5.3. *Encryption*

To provide confidentiality and fine-grained access control, each **DO** determines authorized **DU**s to access its data, by defining an access tree $\mathcal{T}$. Then, it encrypts the data under the access policy by using the following algorithm (see Fig. 8):

**Enc**$(PK, M, \mathcal{T})$: This algorithm selects $r \in \mathbb{Z}_q$ uniformly at random and runs **Share**$(r, q, \mathcal{T}) \rightarrow \{q_{v_a}(0)\}_{v_a \in L_{\mathcal{T}}}$. It outputs:

$$CT = (\mathcal{T}, V = M.g_0^r, V' = g_1^r, C = rP_2, \left\{C_a = q_{v_a}(0)P_0, C'_a = q_{v_a}(0)(PK_a - P_2)\right\}_{v_a \in L_{\mathcal{T}}}). \tag{13}$$

## 5.4. *Decryption*

In this phase, to decrypt a ciphertext $CT$, an authorized **DU** runs **TokenGen**$(PK, CT, SK_{\mathbf{u}})$ algorithm and generates a decryption token $TK_{CT}^{(\mathbf{u})}$ and its associated private-key $d$. It gives $TK_{CT}^{(\mathbf{u})}$ to the **CSP** and keeps $d$ confidential. The **CSP** runs **PartDec**$(PK, CT, TK_{CT}^{(\mathbf{u})})$ algorithm and returns a partiality decrypted ciphertext $CT'$ to the **DU**. Finally, by using the private-key $d$ and partially decrypted ciphertext $CT'$, the **DU** runs **FullDec**$(PK, CT, d, CT')$ algorithm and recovers the associated message (see Fig. 9). The three algorithms are defined as follows:

**TokenGen**$(PK, CT, SK_{\mathbf{u}})$: Consider a ciphertext $CT = (\mathcal{T}, V, V', C, \left\{C_a, C'{}_a\right\}_{v_a \in L_{\mathcal{T}}})$ and a **DU**'s secret-key set $SK_{\mathbf{u}}$ associated with an attribute set $Att$. If there is an attribute set $\{a_j\}_{j=1}^t \subseteq Att$ satisfying $\mathcal{T}$, this algorithm outputs $(d, TK_{CT}^{(\mathbf{u})})$, where $d$ is a uniform element of $\mathbb{Z}_q$ and

$$TK_{CT}^{(\mathbf{u})} = \{T_{a_j}^{(\mathbf{u})} = SK_{i_j, a_j, u} + dP_2\}_{j=1}^t. \tag{14}$$

Otherwise, it outputs $\perp$.

**PartDec**$(PK, CT, TK_{CT}^{(\mathbf{u})})$: Consider a ciphertext $CT = (\mathcal{T}, V, V', C, \left\{C_a, C'{}_a\right\}_{v_a \in L_{\mathcal{T}}})$, a **DU** with identifier $ID_{\mathbf{u}}$, and a token $TK_{CT}^{(\mathbf{u})} = \{T_{a_j}^{(\mathbf{u})}\}_{j=1}^t$. If $\{a_j\}_{j=1}^t$ does not satisfy $\mathcal{T}$, then this algorithm outputs $\perp$. Otherwise it computes:

$$B_{v_{a_j}, d, \mathbf{u}} = \frac{\hat{e}(C_{a_j}, T_{a_j}^{(u)})}{\hat{e}(C_{a_j}, PK^{(i_j)})\hat{e}(C'_{a_j}, ID_u)}$$

$$= g_0^{q_{v_{a_j}}(0)}.g_1^{q_{v_{a_j}}(0)}.\hat{e}(P_2, ID_{\mathbf{u}})^{q_{v_{a_j}}(0)}.\hat{e}(P_0, dP_2)^{q_{v_{a_j}}(0)}, \tag{15}$$

for any $j \in \{1, \dots, t\}$. Then, it runs **Combine**$(q, \mathcal{T}, \{B_{v_{a_j},d,\mathbf{u}}\}_{j=1}^{t})$ and obtains

$$B_{R,d,\mathbf{u}} = g_0^r.g_1^r.\hat{e}(P_2, ID_{\mathbf{u}})^r.\hat{e}(P_0, dP_2)^r. \tag{16}$$

Afterwards, it returns the partially decrypted ciphertext

$$\begin{aligned} CT' &= \frac{B_{R,d,\mathbf{u}}}{V'.\hat{e}(C, ID_u)} \\ &= g_0^r.\hat{e}(dP_0, C). \end{aligned} \tag{17}$$

**FullDec**$(PK, CT, d, CT')$: Using a private-key $d$ generated by **TokenGen**$(PK, CT, SK_{\mathbf{u}})$ and partially decrypted ciphertext $CT'$ generated by **PartDec**$(PK, CT, TK_{CT}^{(\mathbf{u})})$, this algorithm decrypts a ciphertext $CT = (\mathcal{T}, V, V', C, \{C_a, C'_a\}_{v_a \in L_{\mathcal{T}}})$ as follows:

$$M = \frac{V}{CT'.\hat{e}(-dP_0, C)}. \tag{18}$$

**Remark 3.** A **DU** with an attribute set $S$ satisfying the access tree of a ciphertext $CT = (\mathcal{T}, V, V', C, \{C_a, C'_a\}_{y_a \in L_{\mathcal{T}}})$ can decrypt $CT$ without the participation of the **CSP** as follows:

- At first, it calculates

$$\begin{aligned} B_{v_{a_j},\mathbf{u}} &= \frac{\hat{e}(C_{a_j}, SK_{i_j,a_i,\mathbf{u}})}{\hat{e}(C_{a_j}, PK_{i_j}).\hat{e}(C'_{a_j}, ID_{\mathbf{u}})} \\ &= g_0^{q_{v_{a_j}}}.g_1^{q_{v_{a_j}}}.\hat{e}(P_2, ID_{\mathbf{u}}))^{q_{v_{a_j}}}, \end{aligned} \tag{19}$$

  for each attribute $a_j \in S$.
- Then, it runs **Combine**$(q, \mathcal{T}, \{B^d_{v_{a_j},\mathbf{u}}\}_{j=1}^{t})$ algorithm and obtains $B_{R,\mathbf{u}} = g_0^r.g_1^r.\hat{e}(P_2, ID_{\mathbf{u}}))^r$. Considering,

$$\begin{aligned} B_{R,u} &= g_0^r.g_1^r.\hat{e}(P_2, ID_{\mathbf{u}}))^r \\ &= g_0^r.g_1^r.\hat{e}(rP_2, ID_{\mathbf{u}}) \\ &= g_0^r.V'.\hat{e}(C, ID_{\mathbf{u}}), \end{aligned} \tag{20}$$

  it computes:

$$M = \frac{V}{\frac{B_{R,\mathbf{u}}}{V'.\hat{e}(C, ID_{\mathbf{u}})}}. \tag{21}$$

**Remark 4.** One of the most important benefits of our FDR-CP-HABE can be observed in **Decryption** phase. As we have seen in this section, in our scheme, secret-keys of a **DU** issued by different **DA**s can be used together in **Decryption** phase. This option offers a high level of scalability and flexibility in **Key delegation** Phase.

### *5.5. User revocation*

In real applications, some attributes may be revoked from a **DU**. To achieve forward secrecy, whenever a **DU** loses an attribute, it should be made sure that it no longer has a valid secret-key corresponding to the attribute. To fulfill this requirement, once a domain **DA**$_i$ judges that a **DU** has lost an attribute $a_0$, it runs **UpdateAtt**$(PK, a_0, MSK^{(i)})$ algorithm and updates the master secret-key and public parameters of the system and generates an update-key $UK_{a_0}$ corresponding to the new master secret-key. The identity of the revoked user and the generated update-key are given to **DA**s that have the attribute, the new public parameter is published to the system, and also the update-key is sent to the **CSP**. Then, each **DU** possessing the attribute with identifier $ID_{\mathbf{u}}$ should request a **DA** supporting the attribute to update its secret-key. The **DA** runs **UpdateKey**$(PK, a_0, UK_{a_0}, SK_{i,a_0,\mathbf{u}}, ID_{\mathbf{u}})$ algorithm and returns the updated secret-key. Also, using $UK_{a_0}$, the **CSP** runs **ReEnc**$(PK, a_0, UK_{a_0}, CT)$ algorithm and re-encrypts the outsourced ciphertexts that their access trees have a leaf node corresponding to the attribute (see Fig. 10). The mentioned three algorithms are defined as follows:

**UpdateAtt**$(PK, a_0, MSK^{(i)})$: It selects $sk'_{a_0} \in \mathbb{Z}_q$ uniformly at random and sets $PK'_{a_0} = sk'_{a_0} P_0$ and

$$UK_{a_0} = sk'_{a_0} - sk_{a_0}. \tag{22}$$

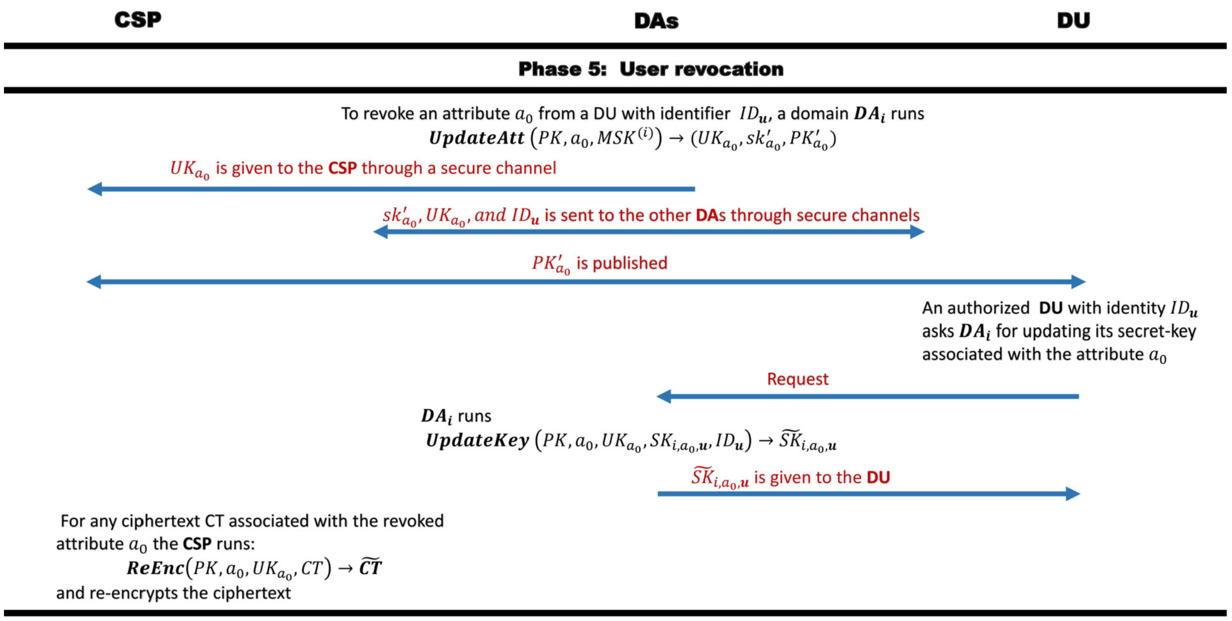

CSP
DAs
DU
Phase 5: User revocation
To revoke an attribute $a_0$ from a DU with identifier $ID_{\mathbf{u}}$, a domain $DA_i$ runs
$UpdateAtt(PK, a_0, MSK^{(i)}) \rightarrow (UK_{a_0}, sk'_{a_0}, PK'_{a_0})$
$UK_{a_0}$ is given to the CSP through a secure channel
$sk'_{a_0}, UK_{a_0}$, and $ID_{\mathbf{u}}$ is sent to the other DAs through secure channels
$PK'_{a_0}$ is published
An authorized DU with identity $ID_{\mathbf{u}}$ asks $DA_i$ for updating its secret-key associated with the attribute $a_0$
Request
$DA_i$ runs
$UpdateKey(PK, a_0, UK_{a_0}, SK_{i,a_0,\mathbf{u}}, ID_{\mathbf{u}}) \rightarrow \widetilde{SK}_{i,a_0,\mathbf{u}}$
$\widetilde{SK}_{i,a_0,\mathbf{u}}$ is given to the DU
For any ciphertext CT associated with the revoked attribute $a_0$ the CSP runs:
$ReEnc(PK, a_0, UK_{a_0}, CT) \rightarrow \widetilde{CT}$
and re-encrypts the ciphertext

**Fig. 10.** Procedures of Phase 5 of the scheme.

**UpdateKey**$(PK, a_0, UK_{a_0}, SK_{i,a_0,\mathbf{u}}, ID_{\mathbf{u}})$: Given an update-key $UK_{a_0}$ and a **DU** with identifier $ID_{\mathbf{u}}$, this algorithm updates a secret-key $SK_{i,a_0,\mathbf{u}}$ of the **DU** as follows:

$$\widetilde{SK}_{i,a,\mathbf{u}} = SK_{i,a,\mathbf{u}} + UK_{a_0} ID_{\mathbf{u}}. \tag{23}$$

**ReEnc**$(PK, a_0, UK_{a_0}, CT)$: For a revoked attribute $a_0$, the **CSP** re-encrypts any ciphertext $CT = (\mathcal{T}, V, V', C, \left\{C_a, C'_a\right\}_{v_a \in L_{\mathcal{T}}})$ that $\mathcal{T}$ has a leaf node $v_{a_0}$ corresponding to $a_0$ as

$$\widetilde{CT} = (\mathcal{T}, V, C, \{C_{a_0}, \widetilde{C}'_{a_0} = C'_{a_0} + UK_{a_0} C_{a_0}\} \cup \{C_a, C'_a\}_{v_a \in L_{\mathcal{T}} - \{v_{a_0}\}}). \tag{24}$$

**Remark 5.** In **User revocation** phase, one can see another benefit of our proposed scheme. In this phase, the parameters associated with the revoked attributes and also secret-keys of authorized **DU**s can be updated by any **DA** supporting the attribute even if the secret-key have not been issued by the **DA**.

### 5.6. *Correctness analysis of our proposed scheme*

In the following, we analyze the correctness of our proposed scheme.

**Theorem 6.** *The* ***Decryption*** *phase is correct.*

**Proof.** The proof is given in the Appendix. □

**Theorem 7.** *The* ***ReEnc*** *algorithm is correct.*

**Proof.** This theorem is proved in the Appendix. □

## 6. Security analysis

In this section, we show that our proposed scheme fulfills the security requirements mentioned in Section 4. We first prove that it is CPA-secure in the standard model. Then, we conclude that it is collusion resistant and achieves fine-grained access control over outsourced encrypted data.

### 6.1. *Data confidentiality*

**Theorem 8.** *If the* ***DBDH*** *problem is hard relative to* $\mathcal{G}$*, then our construction is CPA-secure in the standard model.*

**Proof.** Let $\Pi$ denote our proposed scheme, and let $\mathcal{A}$ be a PPT adversary in the experiment $\mathbf{HABE}^{cpa}_{\mathcal{A},\Pi}(n)$ introduced in Section 4.7, where $n$ is the security parameter of the system. In the following, we show that there is a negligible function *negl* such that

$$\Pr(\mathbf{HABE}^{cpa}_{\mathcal{A},\Pi}(n)=1) \leq \frac{1}{2} + negl(n). \tag{25}$$

Consider a PPT distinguisher $\mathcal{D}$ attempting to solve the **DBDH** problem. The distinguisher $\mathcal{D}$ receives a tuple $(q, G_1, G_2, \hat{e}, P, \alpha P, \beta P, \gamma P, h = \hat{e}(P, P)^z)$ from a challenger, where $(q, G_1, G_2, \hat{e})$ is an output of $\mathcal{G}(1^n)$, $\alpha, \beta, \gamma$ are three uniform elements of $\mathbb{Z}_q$, $P$ is a random generator of $G_1$, and $z$ is either equal to $\alpha\beta\gamma$ or is a uniform element of $\mathbb{Z}_q$. $\mathcal{D}$ aims to determine the case of $z$. It runs $\mathcal{A}$ as a subroutine as follows:

1. **Setup**: $\mathcal{D}$ selects two random elements $t_1, t_2 \in \mathbb{Z}_q$ and sets

$$P_0 = P \tag{26}$$

$$P_1 = \gamma P \tag{27}$$

$$Q_0 = \alpha P \tag{28}$$

$$Q_1 = \gamma P - t_1 P = (\gamma - t_1)P \tag{29}$$

$$P_2 = t_2 P_0 \tag{30}$$

$$g_0 = \hat{e}(Q_0, P_1) \tag{31}$$

$$g_1 = \hat{e}(Q_1, -Q_0). \tag{32}$$

Then, it considers an attribute set $\mathbb{U}$ as the universal attribute set and chooses $sk_a \in \mathbb{Z}_q$ uniformly at random, for any $a \in \mathbb{U}$. It gives the system public parameters $PK = (q, G_1, G_2, \hat{e}, n, P_0, P_1, P_2, Q_0, Q_1, g_0, g_1, \{PK_a\}_{a\in\mathbb{U}})$ to the adversary $\mathcal{A}$. Then, $\mathcal{D}$ selects a random element $X'$ and assumes that for an unknown value $X \in G_1$, $X'$ satisfies

$$X' = \alpha\gamma P + X. \tag{33}$$

Also, $\mathcal{D}$ assumes that

$$MSK = (X, sk_0 = \alpha, x = \gamma - t_1, SK_1 = \alpha\beta P, SK_2, \{sk_a\}_{a\in\mathbb{U}}, \{K_a\}_{a\in\mathbb{U}}) \tag{34}$$

is the master secret-key corresponding to $PK$. Note that the value of $X, sk_0, x, SK_1$, and $SK_2$ in (34) are unknown to $\mathcal{D}$. For an integer $k$, $\mathcal{D}$ selects $k$ identifiers $\{ID_i\}_{i=1}^k$ and $k$ random elements $\{sk_i\}_{i=1}^k \subset \mathbb{Z}_q$. Then, it computes:

$$SK^{(i)} = X' + sk_i ID_i \tag{35}$$

and

$$PK^{(i)} = X' - t_1\alpha P + sk_i ID_i. \tag{36}$$

For $i = 1, \ldots, k$, it gives $ID_i$ and $PK^{(i)}$ to $\mathcal{A}$ as the identifier and the public-key of the $i$-th domain authority $\mathbf{DA}_i$, respectively.

**Remark 9.** In the construction introduced in Section 5, we had $Q_1 = xP_0$, $Q_0 = sk_0 P_0$, $SK_1^1 = sk_0 P_1$ and $SK_1^2 = xQ_0 = xsk_0 P_0$, where $x$ and $sk_0$ are two random elements of $\mathbb{Z}_q$. So, considering $Q_1 = (\gamma - t_1)P_0$, $Q_0 = \alpha P_0$ and $P_1 = \gamma P_0$, one gets:

$$x = \gamma - t_1,\ sk_0 = \alpha. \tag{37}$$

Therefore, we have:

$$SK_1 = sk_0 P_1 = \alpha\gamma P_0, \tag{38}$$

and

$$SK_2 = xQ_0 = xsk_0 P_0 = (\gamma - t_1)\alpha P_0 = \gamma\alpha P_0 - t_1\alpha P_0. \tag{39}$$

Combining (33) and (38), one concludes:

$$\begin{aligned} SK^{(i)} &= X' + sk_i ID_i \\ &= \alpha\gamma P + X + sk_i ID_i \\ &= SK_1 + X + sk_i ID_i. \end{aligned} \tag{40}$$

Also, from Equations (33) and (39), we have:

$$\begin{aligned} PK^{(i)} &= X' - t_1\alpha P + sk_i ID_i \\ &= \alpha\gamma P + X - t_1\alpha P + sk_i ID_i \\ &= \gamma\alpha P - t_1\alpha P + X + sk_i ID_i \\ &= SK_2 + X + sk_i ID_i. \end{aligned} \tag{41}$$

Therefore, $SK^{(i)}$ and $PK^{(i)}$ are chosen correctly.

2. **Phase 1**: When the adversary $\mathcal{A}$ requests the secret-key corresponding to $(ID_i, ID_\mathbf{u}, a)$ from $\mathcal{O}_{\mathbf{KeyGen}}$, the distinguisher $\mathcal{D}$ returns

$$SK_{i,a,\mathbf{u}} = SK^{(i)} + sk_a ID_\mathbf{u} \tag{42}$$

and adds $a$ to $L_{ID_\mathbf{u}}$ and also puts $ID_\mathbf{u}$ into $L_{B_2}$.

3. **Challenge**: Adversary $\mathcal{A}$ outputs two equal length plaintexts $M_0$ and $M_1$, and an access tree $\mathcal{T}$ such that for any **DU** with identifier $ID_\mathbf{u}$, $L_{ID_\mathbf{u}}$ does not satisfy $\mathcal{T}$. When $\mathcal{D}$ receives the plaintexts, it chooses $b \in \{0,1\}$ and encrypts $M_b$ under $\mathcal{T}$ as follows:
Firstly, $k_R$ elements $b_1 \dots b_{k_R} \in \mathbb{Z}_q$ are chosen uniformly at random, where $k_R$ is the threshold value of the root node of $\mathcal{T}$. Then, it calculates $sk_a(\beta P_0) - t_2(\beta P_0) = \beta(sk_a P_0 - t_2 P_0) = \beta(PK_a - P_2)$ and considers two following polynomials:

$$\begin{cases} Q_R : \mathbb{Z}_q \to G_1 \\ Q_R(x) = \beta P_0 + (b_1 P_0)x + \dots + (b_{k_R-1} P_0)x^{k_R-1} \end{cases} \tag{43}$$

and

$$\begin{cases} Q'_R : \mathbb{Z}_q \to G_1 \\ Q'_R(x) = \beta(PK_a - P_2) + b_1(PK_a - P_2)x + \dots + b_{k_R-1}(PK_a - P_2)x^{k_R-1}. \end{cases} \tag{44}$$

It generates two polynomials

$$\begin{cases} Q_{c_i} : \mathbb{Z}_q \to G_1 \\ Q_{c_i}(x) = Q_R(i) + (b_1^{(i)} P_0)x + \dots + (b_{k_{c_i}-1}^{(i)} P_0)x^{k_{c_i}-1} \end{cases} \tag{45}$$

and

$$\begin{cases} Q'_{c_i} : \mathbb{Z}_q \to G_1 \\ Q'_{c_i}(x) = Q'_R(i) + (b_1^{(i)}(PK_a - P_2))x + \dots + (b_{k_{c_i}-1}^{(i)}(PK_a - P_2))x^{k_{c_i}-1}, \end{cases} \tag{46}$$

for each $i$-th child $c_i$ of the root node, in a similar way.
By continuing this process, two $(k_v - 1)$-degree polynomials $Q_v$ and $Q'_v$ are generated for any node $v$ in $\mathcal{T}$. Since, the threshold value of each leaf node is equal to one, the polynomials corresponding to any leaf node $v_a$ are two constant polynomials $Q_{v_a}, Q'_{v_a} \in G_1$. Finally, $\mathcal{D}$ outputs

$$CT_b = (\mathcal{T}, V = M_b.h, V' = h^{-1}.\hat{e}(\alpha P_0, \beta P_0)^{t_1}, C = t_2(\beta P_0) = \beta(t_2 P_0) = \beta P_2, \left\{C_a = Q_{v_a}, C'_a = Q'_{v_a}\right\}_{v_a \in L_\mathcal{T}}), \tag{47}$$

4. **Phase 2:** $\mathcal{A}$ submits some queries to the following oracles:
$\mathcal{O}_{\mathbf{KeyGen}}(ID_i, ID_\mathbf{u}, a)$: The challenger checks if $L_{ID_\mathbf{u}} \cup \{a\}$ satisfies $\mathcal{T}$ or not. If not, it answers the query the same as before. Otherwise, it aborts.
$\mathcal{O}_{\mathbf{TokenGen}}(CT_b, ID_\mathbf{u}, Att)$: The challenger first checks whether $ID_u \in L_{B_2}$ or not. If so, it aborts. Otherwise, it generates $SK_\mathbf{u} = \{SK_{i,a,u}\}_{a \in Att}$ as before and returns $TK_{CT_b}^{(\mathbf{u})} = \{SK_{i,a,u} + dP_2\}_{a \in Att}$ to the adversary, where $d \in \mathbb{Z}_q$ is chosen uniformly at random. Finally, it adds $ID_\mathbf{u}$ into $L_{B_1}$.
5. **Guess**: $\mathcal{A}$ outputs $b' \in \{0,1\}$, and $\mathcal{D}$ checks whether $b' = b$ or not. If so, $\mathcal{D}$ outputs 1. Otherwise, it returns 0.

Suppose that the randomness $r \in \mathbb{Z}_q$ chosen in $\mathbf{Enc}(PK, M_b, \mathcal{T})$ algorithm is equal to $\beta$. So, from (13), the generated ciphertext should be as follows:

$$(\mathcal{T}, V = M_b.\hat{e}(Q_0, P_1)^{\beta}, V' = \hat{e}(Q_1, -Q_0)^{\beta}, C = \beta P_2, \left\{C_a = q_{v_a} P_0, C'_a = q_{v_a}(PK_a - P_2)\right\}_{v_a \in L_{\mathcal{T}}}), \tag{48}$$

where $\{q_{v_a(0)}\}_{v_a \in L_{\mathcal{T}}}$ is an output of $\mathbf{Share}(\beta, q, \mathcal{T})$. Comparing Equations (27), (28) and (29), we see that $V$ and $V'$ should satisfy the following equations:

$$V = M_b.\hat{e}(Q_0, P_1)^{r} = M_b.\hat{e}(\alpha P, \gamma P)^{\beta} = M_b.\hat{e}(P, P)^{\alpha\beta\gamma} \tag{49}$$

$$V' = \hat{e}(Q_1, -Q_0)^{\beta} = \hat{e}((\gamma - t_1)P, -\alpha P)^{\beta} = \hat{e}(P, P)^{-\alpha\beta\gamma}.\hat{e}(P, P)^{t_1\alpha\beta} \tag{50}$$

So, if $h = \hat{e}(P, P)^{\alpha\beta\gamma}$, then $V$ and $V'$ in (47) are chosen correctly. Also, the correctness of the other components in (47) can easily be verified. Therefore, if $h = \hat{e}(P_0, P_0)^{\alpha\beta\gamma}$, then $CT_b$ is a valid ciphertext of $M_b$ and thus

$$\Pr(\mathcal{D}(P, \alpha P, \beta P, \gamma P, h = \hat{e}(P, P)^{\alpha\beta\gamma}) = 1) = \Pr(\mathbf{HABE}(n)^{cpa}_{\mathcal{A},\Pi} = 1). \tag{51}$$

Moreover, if $h = \hat{e}(P_0, P_0)^{z}$ for a uniform element $z \in \mathbb{Z}_q$, then $V$ is also a uniform element of $G_1$ and therefore the adversary $\mathcal{A}$ cannot obtain any information about $M_b$ and, equivalently,

$$\Pr(\mathcal{D}(P, \alpha P, \beta P, \gamma P, h = \hat{e}(P, P)^{z}) = 1) = \Pr(b = b') = \frac{1}{2}. \tag{52}$$

On the other hand, under the hardness assumption of the **DBDH** problem, we have:

$$|\Pr(\mathcal{D}(P, \alpha P, \beta P, \gamma P, h = \hat{e}(P, P)^{\alpha\beta\gamma}) = 1) - \Pr(\mathcal{D}(P, \alpha P, \beta P, \gamma P, h = \hat{e}(P, P)^{z}) = 1)| \leq negl(n), \tag{53}$$

for a negligible function *negl*. Combining (51), (52) and (53), one concludes that:

$$\Pr(\mathbf{HABE}(n)^{cpa}_{\mathcal{A},\Pi} = 1) \leq \frac{1}{2} + negl(n). \tag{54}$$

This proves the theorem. □

*6.1.1. Collusion resistance*

Our scheme is collusion resistant. Given a ciphertext outsourced to the **CSP**, by Theorem 8, we know that any groups of unauthorized **DU**s can not learn any partial information about the underlying data, and therefore they cannot decrypt the ciphertext.

## *6.2. Fine-grained access control*

Our scheme offers fine-grained access control. As we have seen in Section 5, our proposed scheme enables a **DO** to define a flexible access tree over an attribute set, and, as we have proved in Theorem 8, a **DU** can decrypt the ciphertext if and only if its attribute set satisfies the access tree.

## *6.3. Backward secrecy*

As the distribution of $UK_{a_0} = sk'_{a_0} - sk_{a_0}$ is uniform, the **CSP** cannot learn any partial information about $sk'_{a_0}$ and $sk_{a_0}$. Now, under our assumption that says the **CSP** does not collude with **DU**s, the same as [8,10,42,43], we see that our proposed system gains backward secrecy. That means, once the revocation procedure is done, revoked users lose their access associated with the revoked attributes.

## **7. Performance analysis**

In this section, we first evaluate our proposed scheme by comparing its features to those provided by similar schemes. Then, we compare its execution time and storage cost with some current access control schemes in terms of both theoretical and actual execution overhead. The notations employed in this section are given in Table 2.

The asymptotic computational complexity is measured based on three kinds of operations: paring operation, multiplication operation, and group operation. Also, the asymptotic storage complexity is measured in terms of the bit-length of elements in the two groups $G_1$ and $G_2$. Our implementations are performed on an Ubuntu 18.04 laptop with an Intel Core i5-2410M Processor 2.3 GHz, 6 GB RAM using python Pairing-Based Cryptography (pyPBC) library and Type A pairings [44]. In that library, Type A pairings are constructed on the curve $E(\mathbb{F}_p) : y^2 = x^3 + x$ over the field $\mathbb{F}_p$, where $p$ is a prime number that $p = 3 \bmod 4$ [45]. Let $(q, G_1, G_2, \hat{e})$ be the same as in Section 3, in this case, $q$ is a factor of $p + 1$, $G_1$ and $G_2$ are $q-$order subgroups of points of $E(\mathbb{F}_p)$ and $E(\mathbb{F}_{p^2})$, respectively, and $\hat{e} : G_1 \times G_1 \to G_2$ is a bilinear map.

**Table 2**
Notations used in our numerical comparison.

| Notation | Description |
|---|---|
| $N$ | Number of attributes in an And-gate access control policy |
| $n$ | Security parameter of the system |
| $l_{G_1}$ | Bit length of elements in $G_1$ |
| $l_{G_2}$ | Bit length of elements in $G_2$ |
| $\tau_p$ | Pairing operation time |
| $\tau_e$ | Exponentiation operation time (calculating $g^k$, for a given $k \in \mathbb{Z}_q$ and $g \in G_2$) |
| $\tau_m$ | Multiplication operation time (calculating $rP$, for a given $r \in \mathbb{Z}_q$ and $P \in G_1$) |
| $\tau_{o_1}$ | Time of the group operation in $G_1$ |
| $\tau_{o_2}$ | Time of the group operation in $G_2$ |
| $\lvert\mathbb{U}\rvert$ | Universal attribute set cardinality |
| $\lvert\mathbf{DA}\rvert$ | Number of **DA**s in the system |
| $\lvert Att\rvert$ | Number of a **DU**'s attributes |

**Table 3**
A comparison between existing attribute-based access control schemes and our proposed scheme.

| Schemes | Fully distributed | Decryption outsourcing | Security proof | Security model | Standard Model | User revocation |
|---|---|---|---|---|---|---|
| Yu et al. [38] | No | No | Yes | Selective | Yes | Yes |
| Waters et al. [46] | No | No | Yes | Adaptive | Yes | No |
| Wang et al. [8] | No | No | Yes | Adaptive | No | Yes |
| Ruj et al. [42] | No | No | Yes | Selective | No | Yes |
| Hur et al. [47] | No | No | Yes | Selective | No | Yes |
| Yang et al. [43] | No | Yes | Yes | Selective | No | Yes |
| Jung et al. [48] | No | No | Yes | Selective | No | Yes |
| Liu et al. [9] | No | No | No | \ | \ | Yes |
| Deng et al. [13] | No | No | Yes | Adaptive | Yes | No |
| Li et al. [10] | No | Yes | Yes | Adaptive | Yes | Yes |
| Huang et al. [11] | No | Yes | No | \ | \ | No |
| Our scheme | Yes | Yes | Yes | Adaptive | Yes | Yes |

**Table 4**
Comparison of computational complexity.

| Schemes | Data encryption | Data decryption | Key generation |
|---|---|---|---|
| Jung et al. [48] | $\tau_e + 2N\tau_m + \tau_{o_2}$ | $\tau_p + (2N+1)\tau_m + \tau_e$ | $(2\lvert Att\rvert + 2\lvert\mathbf{DA}\rvert)\tau_m + (3\lvert\mathbf{DA}\rvert - 1)\tau_{o_1}$ |
| Li et al. [10] | $\tau_e + N(\lvert\mathbf{DA}\rvert + 1)(\tau_m + \tau_{o_2}) + \tau_{o_1}$ | $\tau_e + N\tau_m + \tau_{o_2}$ | $2(\lvert Att\rvert + 2)\tau_m + 2(\lvert Att\rvert + 2)\tau_{o_1}$ |
| Hur et al. [47] | $\tau_e + (2N+1)\tau_m + \tau_{o_2}$ | $N(2\tau_p + \tau_e) + (2N-1)\tau_{o_2}$ | $(2\lvert Att\rvert + 2)\tau_m + \lvert Att\rvert\tau_{o_1}$ |
| Huang et al. [11] | $\tau_p + (2N+1)\tau_m + \tau_e$ | $\tau_p + \tau_{o_2}$ | $(2\lvert Att\rvert + 1)\tau_m + (3\lvert Att\rvert + 1)\tau_{o_1}$ |
| Our scheme | $\tau_e + (2N+1)\tau_m + N\tau_{o_1} + \tau_{o_2}$ | $\tau_p + (N+1)\tau_{o_1} + 2\tau_{o_2}$ | $\lvert Att\rvert\tau_m + \lvert Att\rvert\tau_{o_1}$ |

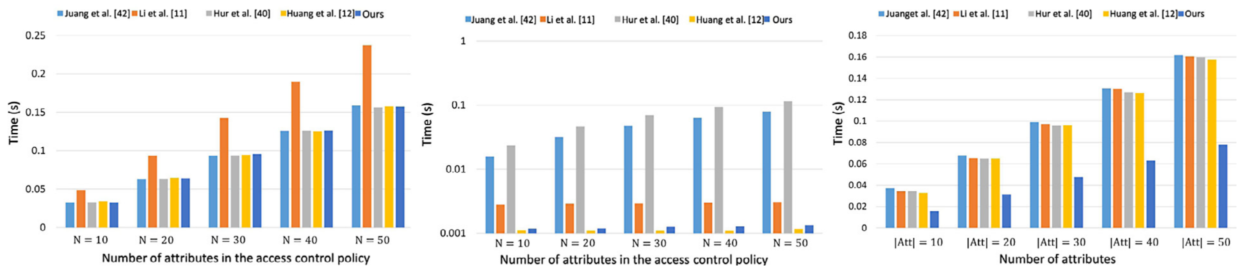
In this section, we assume that bit-lengths of $p$ and $q$ are equal to 512 and 160, respectively. Therefore, under the above assumptions, we have $l_{\mathbb{Z}_q} = 160$ bits, $l_{G_1} = 512$ bits, and $l_{G_2} = 1024$ bits, where $l_G$ denotes the bit-length of elements in group $G \in \{\mathbb{Z}_q, G_1, G_2\}$. Also, we assume that access control policies are in the form of $(a_1\mathbf{AND}\ldots\mathbf{AND}a_K)$ which are simplified versions of access trees.

### *7.1. Comparison*

Table 3 describes features of some existing attribute-based access control systems for cloud computing. One could see that current systems are not fully distributed in their **Key delegation** and **User revocation** phases. Indeed, our scheme is the only one achieving this feature. The schemes presented in [10,11,43] and our scheme are the only ones providing decryption outsourcing. Also, between the provable secure schemes, only [10,46,13] and our scheme are adaptively secure in the standard model.

### *7.2. Computation cost*

Table 4 compares the asymptotic complexities between attribute-based access control schemes presented in [10,11,47,48] and our proposed scheme. From the table, execution times of **Encryption**, **Decryption** and **Key generation** phases in these schemes depend on: 1) the number of attributes in the access control policy associated with the ciphertext, $N$, 2) the number of domain authorities in the system, $\lvert\mathbf{DA}\rvert$, and 3) the number of attributes of **DU**s, $\lvert Att\rvert$. We evaluate execution times of the phases when $N$ and $\lvert Att\rvert$ are in $\{10, 20, \ldots, 50\}$, and $\lvert\mathbf{DA}\rvert = 2$. Fig. 11(a) presents the running time of **Encryption** phase with different number of attributes in the access control policy, Fig. 11(b) describes the computational overhead on **DU**s in **Decryption** phase with different number of attributes in the access control policy, and Fig. 11(c) shows experimental results of generating the secret-keys of **DU**s with different number of attributes.

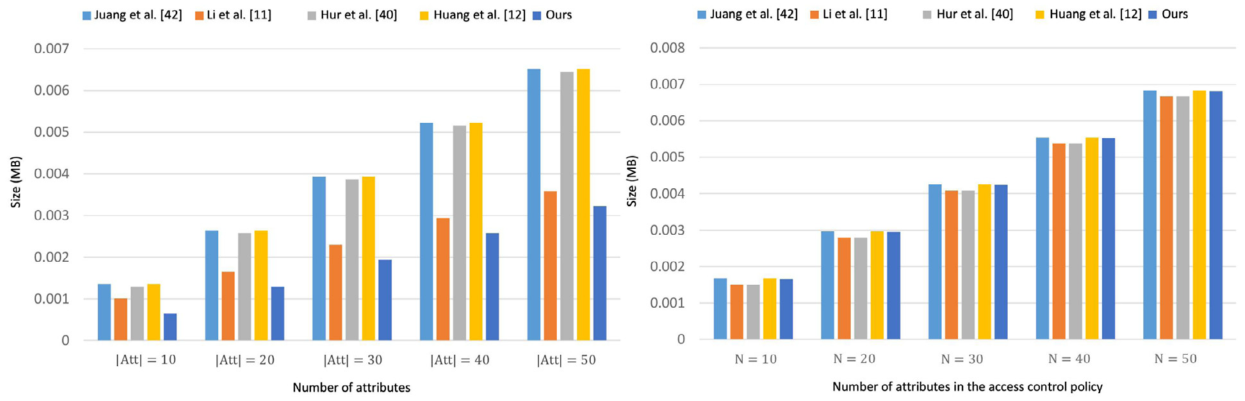


(a) Encryption (b) Decryption (c) Key genration

**Fig. 11.** Comparison of computational overhead in each execution. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

**Table 5**
Storage costs.

| Schemes | Ciphertext | Secret-key |
|---|---|---|
| Jung et al. [48] | $l_{G_2} + l_{G_1}(2N)$ | $(2\|Att\| + 1)l_{G_1}$ |
| Li et al. [10] | $l_{G_2} + l_{G_1}(2N + 1)$ | $(\|Att\| + 5)l_{G_1} + 2l_{\mathbb{Z}_q}$ |
| Hur et al. [47] | $l_{G_2} + l_{G_1}(2N + 1)$ | $2\|Att\|l_{G_1}$ |
| Huang et al. [11] | $2l_{G_2} + (2N + 1)l_{G_1} + n$ | $(2\|Att\| + 1)l_{G_1}$ |
| Our scheme | $2l_{G_2} + (2N + 1)l_{G_1}$ | $\|Att\|l_{G_1}$ |



(a) Secret-key (b) Ciphertext

**Fig. 12.** Size of a **DU**'s secret-key and an outsourced ciphertext. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

The comparisons show that, in our proposed scheme, the performance of **Encryption** phase is almost the same as [11,47,48], and the encryption mechanism in [10] is more expensive than the others. In **Decryption** phase of [11] and our scheme, the decryption-time overhead on **DU**s is significantly less than the others. Also, our proposed **Key generation** mechanism is more efficient than the others.

### 7.3. *Storage cost*

In Table 5, we calculate the storage cost incurred by maintaining encrypted data and data users' secret-keys. As we see in the table, lengths of a ciphertext and a **DU**'s secret-keys are functions of the number of attributes in the access control policy associated with the ciphertext, $N$, and the cardinality of the attribute set of the **DU**, $|Att|$, respectively. In our evaluation, we assume that $|Att|$ and $N$ are in $\{10, 20, \ldots, 50\}$.

Fig. 12(a) and 12(b) describe the length of a ciphertext and the size of a **DU**'s secret-keys with different number of attributes. From the figure, in our scheme, secret-key size of **DU**s is significantly less than size of secret-keys in the other schemes, and size of ciphertexts in these five schemes are almost the same as each other.

## 8. Conclusion

In this paper, we proposed the concept of a fully distributed revocable ciphertext-policy hierarchical attribute-based encryption (FDR-CP-HABE) system, and we proposed the first FDR-CP-HABE scheme. Our proposed scheme offers a high level of flexibility and scalability in key delegation and user revocation mechanisms. Our scheme enables data owners to define and to enforce access control policies on a set of attributes. We proved that the scheme is semantically secure in the standard model based on the hardness assumption of the decisional bilinear Diffie–Hellman (**DBDH**) problem. We also showed that our proposed scheme achieves fine-grained data access control over the outsourced ciphertexts, and it is resistant against the collusion attack of unauthorized data users. Our proposed scheme is efficient and offers lightweight computing in the decryption phase. We analyzed the performance of our proposed scheme and made a comparison between it and some current schemes. We observed that the performance is acceptable compared with the other similar schemes. Considering the security, efficiency, scalability, and flexibility of the scheme, one concludes that it is suitable for cloud computing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A

**Theorem 6.** The **Decryption** phase is correct.

**Proof.** Consider a ciphertext $CT = (\mathcal{T}, V, V', C, \{C_a, C'_a\}_{y_a \in L_{\mathcal{T}}})$. Let $TK^{(\mathbf{u})}_{CT} = \{T^{(\mathbf{u})}_{a_j} = SK_{i_j,a_j,u} + dP_2\}^t_{j=1}$, $d$, $Att$, and $\{a_j\}^t_{j=1} \subseteq Att$ be the same as Section 5.4. In the following, we first show the correctness of **PartDec** algorithm. Then, we conclude that **FullDec** algorithm is correct. To prove Equations (15) and (17), we have:

$$\begin{aligned}
B_{v_{a_j},d,\mathbf{u}} &= \frac{\hat{e}(C_{a_j}, T^{(u)}_{a_j})}{\hat{e}(C_{a_j}, PK^{(i_j)})\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(13),(14)}{=} \frac{\hat{e}(q_{v_{a_j}}(0)P_0, SK_{i_j,a_j,u} + dP_2)}{\hat{e}(C_{a_j}, PK^{(i_j)})\hat{e}(C'_{a_j}, ID_u)} \\
&= \frac{\hat{e}(q_{v_{a_j}}(0)P_0, SK_{i_j,a_j,u}).\hat{e}(q_{v_{a_j}}(0)P_0, dP_2)}{\hat{e}(C_{a_j}, PK^{(i_j)})\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(9),(12)}{=} \frac{\hat{e}(q_{v_{a_j}}(0)P_0, -xQ_0 + xQ_0 + SK_1 + X + sk_{i_j}ID_{i_j} + sk_{a_j}ID_u).\hat{e}(q_{v_{a_j}}(0)P_0, dP_2)}{\hat{e}(C_{a_j}, PK^{(i_j)})\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(8)}{=} \frac{\hat{e}(q_{v_{a_j}}(0)P_0, SK_1).\hat{e}(q_{v_{a_j}}(0)P_0, -xQ_0 + SK_2 + X + sk_{i_j}ID_{i_j}).\hat{e}(q_{v_{a_j}}(0)P_0, sk_{a_j}ID_u).\hat{e}(q_{v_{a_j}}(0)P_0, dP_2)}{\hat{e}(C_{a_j}, PK^{(i_j)}).\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(10),(13)}{=} \frac{\hat{e}(q_{v_{a_j}}(0)P_0, SK_1).\hat{e}(q_{v_{a_j}}(0)P_0, -xQ_0 + PK^{(i_j)}).\hat{e}(q_{v_{a_j}}(0)P_0, sk_{a_j}ID_u).\hat{e}(q_{v_{a_j}}(0)P_0, dP_2)}{\hat{e}(q_{v_{a_j}}(0)P_0, PK^{(i_j)}).\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(7),(8)}{=} \frac{\hat{e}(q_{v_{a_j}}(0)P_0, sk_0P_1).\hat{e}(q_{v_{a_j}}(0)P_0, -xQ_0).\hat{e}(q_{v_{a_j}}(0)(PK_{a_j} - P_2 + P_2), ID_u).\hat{e}(q_{v_{a_j}}(0)P_0, dP_2)}{\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(7),(13)}{=} \frac{\hat{e}(Q_0, P_1)^{q_{v_{a_j}}(0)}.\hat{e}(Q_1, -Q_0)^{q_{v_{a_j}}(0)}.\hat{e}(C'_{a_j}, ID_u).\hat{e}(P_2, ID_u)^{q_{v_{a_j}}(0)}.\hat{e}(P_0, dP_2)^{q_{v_{a_j}}(0)}}{\hat{e}(C'_{a_j}, ID_u)} \\
&\overset{(7)}{=} g_0^{q_{v_{a_j}}(0)}.g_1^{q_{v_{a_j}}(0)}.\hat{e}(P_2, ID_u)^{q_{v_{a_j}}(0)}.\hat{e}(P_0, dP_2)^{q_{v_{a_j}}(0)}.
\end{aligned}$$

This proves Equation (15). Moreover,

$$\begin{aligned}
CT' &= \frac{B_{R,d,\mathbf{u}}}{V'.\hat{e}(C, ID_u)} \\
&\overset{(16)}{=} \frac{g_0^r.g_1^r.\hat{e}(P_2, ID_{\mathbf{u}})^r.\hat{e}(P_0, dP_2)^r}{V'.\hat{e}(C, ID_u)} \\
&\overset{(13)}{=} \frac{g_0^r.V'.\hat{e}(C, ID_{\mathbf{u}}).\hat{e}(dP_0, C)}{V'.\hat{e}(C, ID_u)} \\
&= g_0^r.\hat{e}(dP_0, C).
\end{aligned} \tag{55}$$

Therefore, Equation (17) is correct. Also,

$$\frac{V}{CT'.\hat{e}(-dP_0,C)} \overset{(13),(17)}{=} \frac{M.g_0^r}{g_0^r.\hat{e}(dP_0,C).\hat{e}(-dP_0,C)} = M. \tag{56}$$

This proves Equation (18). □

**Theorem 7**. The **ReEnc** algorithm is correct.

**Proof.** Let $a_0$, $C_{a_0}$, $C'_{a_0}$, $\widetilde{C}'_{a_0}$, $UK_{a_0}$, $sk_{a_0}$, $sk'_{a_0}$, $PK_{a_0}$ and $PK'_{a_0}$ be the same as Section 5.5. To show the correctness of the re-encryption algorithm, we prove that $\widetilde{C}'_{a_0} = q_{y_{a_0}}(PK'_{a_0} - P_2)$. We have:

$$\begin{aligned}
\widetilde{C}'_{a_0} &= C'_{a_0} + UK_{a_0}C_{a_0} \\
&\overset{(13),(22)}{=} q_{y_{a_0}}(PK_{a_0} - P_2) + (sk'_{a_0} - sk_{a_0})(q_{y_{a_0}}P_0) \\
&\overset{(7)}{=} q_{y_{a_0}}(sk_{a_0}P_0 - P_2) + (sk'_{a_0} - sk_{a_0})(q_{y_{a_0}}P_0) \\
&= q_{y_{a_0}}(sk'_{a_0}P_0 - P_2) \\
&= q_{y_{a_0}}(PK'_{a_0} - P_2).
\end{aligned} \tag{57}$$

So, **ReEnc** algorithm is correct. □

## References

[1] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, Cloud computing—the business perspective, Decis. Support Syst. 51 (1) (2011) 176–189.
[2] J. Webster, An enterprise cloud 'survey of surveys', Forbes [online]. Available: http://www.forbes.com/sites/johnwebster/2016/05/03/an-enterprise-cloud-survey-of-surveys/, 2016.
[3] S. Canard, D.H. Phan, D. Pointcheval, V.C. Trinh, A new technique for compacting ciphertext in multi-channel broadcast encryption and attribute-based encryption, Theor. Comput. Sci. 723 (2018) 51–72.
[4] Y.S. Rao, R. Dutta, Computational friendly attribute-based encryptions with short ciphertext, Theor. Comput. Sci. 668 (2017) 1–26.
[5] S. Xu, G. Yang, Y. Mu, Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation, Inf. Sci. 479 (2019) 116–134.
[6] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, ACM, October 2006, pp. 89–98.
[7] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE Symposium on Security and Privacy, SP'07, IEEE, May 2007, pp. 321–334.
[8] G. Wang, Q. Liu, J. Wu, M. Guo, Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers, Comput. Secur. 30 (5) (2011) 320–331.
[9] Q. Liu, G. Wang, J. Wu, Time-based proxy re-encryption scheme for secure data sharing in a cloud environment, Inf. Sci. 258 (2014) 355–370.
[10] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, D. Chen, Secure, efficient and revocable multi-authority access control system in cloud storage, Comput. Secur. 59 (2016) 45–59.
[11] Q. Huang, Y. Yang, M. Shen, Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing, Future Gener. Comput. Syst. 72 (2017) 239–249.
[12] Z. Wan, J.E. Liu, R.H. Deng, HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing, IEEE Trans. Inf. Forensics Secur. 7 (2) (2012) 743–754.
[13] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, W. Shi, Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts, Inf. Sci. 275 (2014) 370–384.
[14] A. Shamir, Identity-based cryptosystems and signature schemes, in: Workshop on the Theory and Application of Cryptographic Techniques, Springer, Berlin, Heidelberg, August 1984, pp. 47–53.
[15] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in: Annual International Cryptology Conference, Springer, Berlin, Heidelberg, August 2001, pp. 213–229.
[16] B. Waters, Efficient identity-based encryption without random oracles, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, May 2005, pp. 114–127.
[17] H. Hong, X. Liu, Z. Sun, A fine-grained attribute based data retrieval with proxy re-encryption scheme for data outsourcing systems, Mob. Netw. Appl. (2018) 1–6.
[18] G. Chunpeng, Z. Liu, J. Xia, F. Liming, Revocable identity-based broadcast proxy re-encryption for data sharing in clouds, IEEE Trans. Dependable Secure Comput. (2019).
[19] D. He, Y. Zhang, D. Wang, K.K.R. Choo, Secure and efficient two-party signing protocol for the identity-based signature scheme in the IEEE P1363 standard for public key cryptography, IEEE Trans. Dependable Secure Comput. (2018).
[20] Y. Zhang, J. Yu, R. Hao, C. Wang, K. Ren, Enabling efficient user revocation in identity-based cloud storage auditing for shared big data, IEEE Trans. Dependable Secure Comput. (2018).
[21] A. Hassan, A.A. Omala, M. Ali, C. Jin, F. Li, Identity-based user authenticated key agreement protocol for multi-server environment with anonymity, Mob. Netw. Appl. 24 (3) (2019) 890–902.
[22] J. Horwitz, B. Lynn, Toward hierarchical identity-based encryption, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, April 2002, pp. 466–481.
[23] J.H. Seo, K. Emura, Revocable hierarchical identity-based encryption via history-free approach, Theor. Comput. Sci. 615 (2016) 45–60.

[24] D. Boneh, X. Boyen, E.J. Goh, Hierarchical identity based encryption with constant size ciphertext, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, May 2005, pp. 440–456.
[25] B. Waters, Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions, in: Annual International Cryptology Conference, Springer, Berlin, Heidelberg, August 2009, pp. 619–636.
[26] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, May 2005, pp. 457–473.
[27] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, May 2010, pp. 62–91.
[28] J. Cui, H. Zhou, Y. Xu, H. Zhong, OOABKS: online/offline attribute-based encryption for keyword search in mobile cloud, Inf. Sci. (2019).
[29] Y. Song, Z. Li, Y. Li, J. Li, Attribute-based signcryption scheme based on linear codes, Inf. Sci. 417 (2017) 301–309.
[30] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, H. Li, Practical attribute-based multi-keyword search scheme in mobile crowdsourcing, IEEE Int. Things J. 5 (4) (2017) 3008–3018.
[31] R. Bobba, H. Khurana, M. Prabhakaran, Attribute-sets: a practically motivated enhancement to attribute-based encryption, in: European Symposium on Research in Computer Security, Springer, Berlin, Heidelberg, September 2009, pp. 587–604.
[32] C. Gentry, A. Silverberg, Hierarchical ID-based cryptography, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, Berlin, Heidelberg, December 2002, pp. 548–566.
[33] S. Wang, J. Yu, P. Zhang, P. Wang, A novel file hierarchy access control scheme using attribute-based encryption, Appl. Mech. Mater. (2014).
[34] S. Wang, J. Zhou, J.K. Liu, J. Yu, J. Chen, W. Xie, An efficient file hierarchy attribute-based encryption scheme in cloud computing, IEEE Trans. Inf. Forensics Secur. 11 (6) (2016) 1265–1277.
[35] X. Liu, J. Ma, J. Xiong, G. Liu, Ciphertext-policy hierarchical attribute-based encryption for fine-grained access control of encryption data, Int. J. Netw. Secur. 16 (6) (2014) 437–443.
[36] J. Li, Q. Wang, C. Wang, K. Ren, Enhancing attribute-based encryption with attribute hierarchy, Mob. Netw. Appl. 16 (5) (2011) 553–561.
[37] M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption, IEEE Trans. Parallel Distrib. Syst. 24 (2013) 131–143.
[38] S. Yu, C. Wang, K. Ren, W. Lou, Attribute based data sharing with attribute revocation, in: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ACM, April 2010, pp. 261–270.
[39] F. Liu, W.K. Ng, W. Zhang, S. Han, Encrypted set intersection protocol for outsourced datasets, in: 2014 IEEE International Conference on Cloud Engineering, IEEE, March 2014, pp. 135–140.
[40] S. Kamara, P. Mohassel, M. Raykova, S. Sadeghian, Scaling private set intersection to billion-element sets, in: International Conference on Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, March 2014, pp. 195–215.
[41] A. Abadi, S. Terzis, R. Metere, C. Dong, Efficient delegated private set intersection on outsourced private datasets, IEEE Trans. Dependable Secure Comput. (2017).
[42] S. Ruj, A. Nayak, I. Stojmenovic, DACC: distributed access control in clouds, in: 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, November 2011, pp. 91–98.
[43] K. Yang, X. Jia, K. Ren, B. Zhang, R. Xie, DAC-MACS: effective data access control for multi-authority cloud storage systems, IEEE Trans. Inf. Forensics Secur. 8 (11) (2013) 1790–1801.
[44] The python pairing based cryptography library, https://github.com/debatem1/pypbc.
[45] B. Lynn, PBC Library Manual 0.5.11, 2006.
[46] A. Lewko, B. Waters, Decentralizing attribute-based encryption, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, May 2011, pp. 568–588.
[47] J. Hur, Improving security and efficiency in attribute-based data sharing, IEEE Trans. Knowl. Data Eng. 25 (10) (2011) 2271–2282.
[48] T. Jung, X.Y. Li, Z. Wan, M. Wan, Privacy preserving cloud data access with multi-authorities, in: 2013 Proceedings IEEE INFOCOM, IEEE, April 2013, pp. 2625–2633.