Fine-Grained Access Control System based on Outsourced Attribute-based Encryption

Jin Li¹, Jingwei Li², Xiaofeng Chen³, Chunfu Jia², Wenjing Lou⁴

¹ School of Computer Science, Guangzhou University, P.R. China
² College of Information Technical Science, Nankai University, P.R. China
³ State Key Laboratory of Integrated Service Networks, Xidian University, P.R. China
⁴ Department of Computer Science, Virginia Polytechnic Institute and State University, USA

Abstract. As cloud computing becomes prevalent, more and more sensitive data is being centralized into the cloud for sharing, which brings forth new challenges for outsourced data security and privacy. Attribute-based encryption (ABE) is a promising cryptographic primitive, which has been widely applied to design fine-grained access control system recently. However, ABE is being criticized for its high scheme overhead as the computational cost grows with the complexity of the access formula. This disadvantage becomes more serious for mobile devices because they are difficult to finish basic ABE operations with such constrained resources.

Aiming at tackling the challenge above, we present a generic and efficient way to implement attribute-based access control system through introducing secure outsourcing techniques into ABE. More precisely, two cloud service providers (CSPs), namely key generation-cloud service provider (KG-CSP) and decryption-cloud service provider (D-CSP) are introduced to perform the outsourced key-issuing and decryption on behalf of attribute authority and users respectively. In order to outsource heavy computation to both CSPs without private information leakage, we formulize an underlying primitive called outsourced ABE (OABE) and propose several constructions with outsourced decryption and key-issuing. Finally, extensive experiment demonstrates that with the help of KG-CSP and D-CSP, efficient key-issuing and decryption are achieved in our constructions.

1 Introduction

Cloud computing is an emerging computing paradigm in which IT resources and capacities are provided as services over the Internet while hiding platform and implementation details. Promising as it is, this paradigm also brings forth new challenges for data security and privacy when users outsource sensitive data for sharing on cloud servers, which are likely outside of the same trusted domain of data owners. An approach to mitigate this challenge is the use of encryption. However, whereas encryption assures the confidentiality of the sensitive data
against cloud servers, the use of traditional encryption approaches does not sufficiently support the enforcement of fine-grained access control policies.

Another possible solution is utilizing the well-studied access control technique\cite{28,32,41}. However, traditional access control systems are mostly designed for in-house services and depend greatly on the system itself to enforce authorization policies. Thus, they cannot be applied in cloud computing because users and cloud servers are no longer in the same trusted domain. For the purpose of helping the data owner impose access control over data stored on untrusted cloud servers, a feasible consideration would be encrypting data through certain cryptographic primitives but disclosing decryption keys only to authorized users.

Upon this intuition, another type of solution \cite{18,24,4,36} is based on a per file access control list (ACL) or file group. As the system scales, however, both ACL and file group-based techniques only provide a coarse-grained access control.

Aiming at providing fine-grained access control over encrypted data, a novel public key primitive namely attribute-based encryption (ABE) \cite{35} is introduced in the cryptographic community. For the first time, ABE enables public key-based one-to-many encryption. In ABE system, users’ keys and ciphertexts are labeled with sets of descriptive attributes and access policies respectively, and a particular key can decrypt a ciphertext only if the associated attributes and policy are matched.

Though ABE is a promising primitive to design fine-grained access control system in cloud computing, there are several challenges remained in the application of ABE.

- One of the main drawbacks of ABE is that the computational cost in decryption phase grows with the complexity of the access formula. The drawback appears more serious for resource-constrained users such as mobile devices and sensors because the heavy decryption in ABE may not be independently performed by such users. Therefore, one challenge is \textit{how to reduce the decryption complexity of ABE such that it can be applied to fine-grained access control for users with resource-constrained devices}.

- Beyond decryption, generating user’s private key in existing ABE schemes also requires a great quantity of modular exponentiations. Furthermore, the revocation of any single user in existing ABE requires key-update at authority for remaining users who share his/her attributes. All of these heavy tasks centralized at authority side would make it become the efficiency bottleneck in the whole access control system. Therefore, another challenge is \textit{how to reduce the key-issuing complexity of ABE such that scalable access control can be supported}.

1.1 Contribution Overview

Aiming at tackling the challenges described above, we propose a generic construction of attribute-based access control system under an interesting architecture, in which two cloud service providers (CSPs) namely key generation-cloud service provider (KG-CSP) and decryption-cloud service provider (D-CSP) are involved.
to perform the outsourced heavy tasks for users’ key issuing and file access. With
the help of the CSPs, the computational complexity at both user and attribute
authority sides is reduced. Furthermore, since only small computation is required
at authority side for single user’s private key update, the proposed system is able
to efficiently support user revocation even if utilizing the straightforward ABE
revocation technique (i.e. update private keys for all the other users affected by
the one to be revoked).

The challenge issue in the proposed system is how to outsource the heavy
computation to the CSPs as much as possible but without private information
leakage. Our solution is introducing an underlying primitive namely outsourced
ABE (OABE), which allows expensive tasks to be securely outsourced to CSPs
to relieve computation overhead at local.

We firstly propose a basic OABE construction with outsourced decryption
for access control system. The proposed construction securely reduces \(2d\) pairing
operations in its original ABE scheme \[35\] to nearly two with the help of D-CSP,
where \(d\) is the threshold value predefined. The idea behind our construction is
introducing a default attribute for each user and generating private key on a hy-
brid attribute set including the default attribute and user’s attributes. Then, the
decryption could be performed in two sub-phases: D-CSP decrypts ciphertext
with the private key component for user’s attributes to generate the partially de-
crypted ciphertext; user completely decrypts the partially decrypted ciphertext
with the other private key component for the default attribute. Another trick
utilized is building a “matching” property between the two components (for
default attribute and for user’s attributes) in each private key. Therefore, a curi-
os user cannot forge a valid private key even if colluding with D-CSP. Actually,
our technique provides a feasible way to realize the “piecewise key generation”
property recently introduced in \[34\]. Though we describe the application of our
technique in KP-ABE, it can be easily extended to CP-ABE as well.

Then, aiming at designing efficient access control system, we provide several
OABE constructions with outsourced key-issuing and decryption. As far as we
know, this is the first time considering outsourcing key-issuing and decryption
in ABE simultaneously.

- Our first construction requires only constant computation (nearly two single-
based exponentiations) at attribute authority during key-issuing besides ef-
cient decryption.
- Our second construction provides access control in a fine-grained manner but
remains nearly the same efficiency as previous constructions in this paper.
- A stronger adversary model namely RDoC (refereed delegation of compu-
tation) model is considered and our third construction provides a secure
solution in this sense but only an extra light-weight operation of key com-
bining is added.

1.2 Organization
The rest of this paper is organized as follows. In Section 2 we describe some
preliminaries. In Section 3, we present the architecture and adversary model for
attribute-based access control system. In Section 4, we build block for OABE with outsourced key-issuing and decryption. An efficient access control system based on OABE is described in Section 5. In Section 6, we propose a basic OABE construction with outsourced decryption for access control. Several OABE constructions with outsourced key-issuing and decryption for improved access control are presented in Section 7. In Section 8, an extensive experimental result is provided for demonstrating the efficiency of our main OABE construction. In Section 9, the previous work related to ours is surveyed. Finally, we draw conclusion in Section 10.

2 Preliminary

In this section, we first define the notations used in this paper, review the notion of ABE and then introduce the cryptographic background about the bilinear map and its complexity assumption.

2.1 Notations

The notations used in this paper are listed in TABLE 1.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Attribute authority</td>
</tr>
<tr>
<td>S-CSP</td>
<td>Storage-cloud service provider</td>
</tr>
<tr>
<td>KG-CSP</td>
<td>Key generation-cloud service provider</td>
</tr>
<tr>
<td>D-CSP</td>
<td>Decryption-cloud service provider</td>
</tr>
</tbody>
</table>

Table 1. Notations Used in This Paper

2.2 Attribute-based Encryption

ABE has been widely applied to impose fine-grained access control on encrypted data recently. There are two kinds of ABE having been proposed: key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE). In KP-ABE, the access policy is assigned in private key, whereas, in CP-ABE, it is specified in ciphertext. Without loss of generality, we are able to denote \((I_{\text{enc}}, I_{\text{key}})\) as the input to encryption and key generation of ABE. Accordingly, in CP-ABE scheme, \((I_{\text{enc}}, I_{\text{key}}) = (\mathcal{A}, \omega)\) while that is \((\omega, \mathcal{A})\) in KP-ABE, where \(\omega\) and \(\mathcal{A}\) are attribute set and access structure, respectively. Then, an ABE scheme is consisted of four algorithms below.

- **Setup(\(\lambda\))**: The setup algorithm takes as input – a security parameter \(\lambda\). It outputs the public key \(PK\) and the master key \(MK\).
KeyGen($I_{key}, MK$) : The key extraction algorithm takes as input – an access structure (resp. attribute set) $I_{key}$ and the master key $MK$. It outputs the user’s private key $SK$.

Encrypt($M, I_{enc}$) : The encryption algorithm takes as input – a message $M$ and the attribute set (resp. access structure) $I_{enc}$. It outputs the ciphertext $CT$.

Decrypt($CT, SK$) : The decryption algorithm takes as input – a ciphertext $CT$ which was assumed to be encrypted under the attribute set (resp. access structure) $I_{enc}$ and the private key $SK$ for access structure (resp. attribute set) $I_{key}$. It outputs the message $M$ if $(I_{key}, I_{enc}) = 1$ and the error symbol $\bot$ otherwise, where the predicate $\gamma$ is predefined.

2.3 Bilinear Pairings

Definition 1 (Bilinear Map). Let $G$ and $G_T$ be cyclic groups of prime order $q$, writing the group action multiplicatively. $g$ is a generator of $G$. Let $e : G \times G \rightarrow G_T$ be a map with the following properties:

- Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1, g_2 \in G$, and $a, b \in \mathbb{Z}_q$;
- Non-degeneracy: There exists $g_1, g_2 \in G$ such that $e(g_1, g_2) \neq 1$, in other words, the map does not send all pairs in $G \times G$ to the identity in $G_T$;
- Computability: There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in G$.

2.4 Decisional Bilinear Diffie-Hellman Assumption

Definition 2 (DBDH Problem). The decision Bilinear Diffie-Hellman (DBDH) problem is that, given $g, g^x, g^y, g^z \in G$ for unknown random values $x, y, z \in \mathbb{Z}_q$, and $T \in R \ G_T$, to decide if $T = e(g, g)^{xyz}$.

We say that the $(t, \epsilon)$-DBDH assumption holds in $G$ if no $t$-time algorithm has probability at least $\frac{1}{2} + \epsilon$ in solving the DBDH problem for non-negligible $\epsilon$.

3 Attribute-based Access Control System Model

In this section, we describe the architecture for the attribute-based access control system and define its security model.

3.1 Architecture for the Attribute-based Access Control System

As shown in Fig. 1, the architecture for the attribute-based access control system consists of the following entities:
Fig. 1. Architecture for Attribute-based Access Control System

- **Attribute Authority (AA).** This is a key authority for the attribute set. It is responsible for generating public and private parameters for the system. Furthermore, it is in charge of issuing, revoking, and updating attribute keys for users.

- **Data Owner.** This is a user who owns data files and wishes to outsource them into the external storage server provided by a CSP. It is responsible for defining and enforcing an attribute set (resp. access policy) on its own files by encrypting them under such attribute set (resp. access policy).

- **User.** This is an entity who wants to access an outsourced file. If the user owns an access structure (resp. attribute set) matching the attributes (policy) embedded in the encrypted file, and is not revoked, he/she will be able to decrypt the ciphertext and obtain the file.

- **Storage-Cloud Service Provider (S-CSP).** This is an entity that provides a data outsourcing service. It is in charge of controlling the accesses from outside users to the storing data in storage servers and providing corresponding contents services. In this paper, we assume that S-CSP is always online and has abundant storage capacity and computation power.

- **Key Generation-Cloud Service Provider (KG-CSP).** This is an entity that provides a computing service. It helps AA manage users (including data owner and users) through undertaking the expensive tasks delegated by AA during key-issuing, revoking and updating.

- **Decryption-Cloud Service Provider (D-CSP).** This is an entity that provides a computing service as well. It helps user efficiently access the outsourced files through performing partial decryption on ciphertext.

We give an overview of the attribute-based access control system as follows.

- **System Setup.** Public parameter and master key are initialized for the system and AA keeps the master key as secret information.
– **New User Grant.** When a new user wants to join the system, with the aid of KG-CSP, AA issues an attribute private key to him/her based on his/her attributes.

– **New File Creation.** When a data owner wants to outsource and share a file with some users, he/she encrypts the file to be uploaded under a specified attribute set (resp. access policy).

– **File Access.** When a user wants to access an outsourced file, he/she downloads ciphertext from S-CSP and decrypts it with the help of D-CSP.

– **User Revocation.** When there is a user to be revoked, AA updates “affected” users’ private keys with the help of KG-CSP, while the “affected” ciphertexts having been stored on S-CSP will be updated as well.

### 3.2 Adversary Model and Security Requirements

We assume that S-CSP, D-CSP and KG-CSP are semi-trusted. More precisely, they will follow our proposed protocols, but try to find out as much secret information as possible based on their possessions. Furthermore, D-CSP is allowed to collude with curious users and S-CSP. Thus, two types of adversaries are considered in our access control system: i) curious users colluding with D-CSP and S-CSP; ii) the curious KG-CSP.

The security requirements considered in this paper are provided as follows.

– **Data Confidentiality.** Unauthorized users without appropriate access structure (resp. attributes) matching the attributes (resp. policy) embedded in ciphertext should be prevented from accessing the underlying plaintext. Additionally, KG-CSP is not fully trusted in the system, it is allowed to perform aided key generation without learning any useful information about users’ private keys.

– **Collusion-Resistance.** Even if a group of users collude with S-CSP and D-CSP, they cannot decrypt a ciphertext by combining their access structures (resp. attributes) if none of the users is unable to decrypt the ciphertext alone.

### 4 Building Block: OABE

In this section, we firstly provide the definition for OABE (outsourced ABE) with outsourced key-issuing and decryption. Then, its security definitions are presented.

#### 4.1 Definition

Based on the system model provided in Section 3, we attempt to define an underlying primitive namely OABE with outsourced key-issuing and decryption for realizing our access control system. Notice that the definitions of Setup and Encrypt in OABE are identical to traditional ABE shown in Section 2.2, we only
show the definitions of *outsourced key-issuing protocol* and *outsourced decryption protocol*.

### Outsourced Key-Issuing Protocol

Three entities including users, AA and KG-CSP are involved in this protocol. Upon receiving a key-issuing request from a user, AA firstly sends an outsourcing key (denoted as $OK$) to KG-CSP and receives a private key component (denoted as $SK_1$) for the user. The other component $SK_2$ is computed locally by AA. At a high level, the protocol is described as follows.

$$\text{User} \xrightarrow{\text{request}} \text{AA} \xrightarrow{OK} \text{KG-CSP}$$

The outsourced key-issuing protocol consists of the following three polynomial-time algorithms.

- **O-KeyGen-PreProc($I_{\text{key}}, MK$)**: The preprocessing algorithm run by AA takes as input – the access structure (resp. attribute set) $I_{\text{key}}$ for a user, the master key $MK$. It outputs the key pair ($OK$, $AK$) where $OK$ denotes the outsourcing key for KG-CSP and $AK$ denotes the secret key for AA to compute the other component of private key.

- **O-KeyGen-Outsource($I_{\text{key}}, OK$)**: The outsourced algorithm run by KG-CSP takes as input – the access structure (resp. attribute set) $I_{\text{key}}$ and the outsourcing key $OK$. It outputs the private key component $SK_1$.

- **O-KeyGen-PostProc($AK, SK_1$)**: The postprocessing algorithm run by AA takes as input – the secret key $AK$ and the private key $SK_1$. It outputs $SK = (SK_1, SK_2)$ as the user’s private key.

### Outsourced Decryption Protocol

Two entities including user and D-CSP are involved in this protocol. More precisely, upon receiving the ciphertext $CT$, the user delivers $SK_1$ along with $CT$ to D-CSP and receives a partially decrypted ciphertext $CT'$. Finally, the message is completely computed by the user with $SK$. At a high level, it can be described as follows.

$$\text{User} \xrightarrow{(CT,SK_1)} \text{D-CSP}$$

The outsourced decryption protocol consists of the following two polynomial-time algorithms.

- **O-Decrypt-Outsource($CT, SK_1$)**: The outsourced algorithm run by D-CSP takes as input – a ciphertext $CT$ assumed to be encrypted under the attribute set (resp. access structure) $I_{\text{enc}}$ and the private key component $SK_1$ for access structure (resp. attribute set) $I_{\text{key}}$. It outputs the partially decrypted ciphertext $CT'$ if $\gamma(I_{\text{key}}, I_{\text{enc}}) = 1$, otherwise outputs $\bot$.

- **O-Decrypt-Dec($CT', SK$)**: The complete decryption algorithm run by the user takes as input – the partially decrypted ciphertext $CT'$ and the private key $SK$. It outputs a message $M$. 

4.2 Security Definition

Based on the two types of adversaries classified in Section 3.2, we provide a more precise consideration in OABE:

- **Type-I Adversary.** It is defined as a curious user colluding with D-CSP. Such an adversary is allowed to ask for all the SK1 and the private keys SK of dishonest users. The goal of this adversary is to obtain useful information from ciphertext not intended for him/her. Notice that Type-I adversary cannot get outsourcing key OK for any user.

- **Type-II Adversary.** It is defined as a curious KG-CSP. Such an adversary owns outsourcing keys OK for all users in the system and tries to extract any useful information from ciphertext.

Having the intuition above, we are able to follow the replayable CCA (RCCA) security given in [8][21] and define RCCA security for both type-I and type-II adversaries in our OABE. Since the security definition is similar to the previous work [21], we just elaborate them in detail in Appendix.

**Definition 3 (RCCA Security).** An OABE scheme with outsourced key-issuing and decryption is secure against replayable chosen-ciphertext attack if all polynomial-time adversaries have at most a negligible advantage in the RCCA security games for both type-I and type-II adversaries.

5 OABE-based Access Control System

In this section, we provide a generic construction of the attribute-based access control system. Its security analysis is presented as well.

5.1 Generic Construction

**System Setup.** Choose a security parameter 1^λ and run the algorithm Setup(1^λ) of OABE to obtain the public parameter PK and the master key MK. The public parameter is then published, while the master key is kept by AA as a secret.

**New File Creation.** Whenever a data owner wants to create and upload a file F to S-CSP, he/she firstly defines an attribute set (resp. access structure) I_enc for this file. Then, the owner randomly picks a symmetric key K from the key space and encrypts the file F with K using standard symmetric key algorithm such as AES to obtain the ciphertext CT_F. Later on, he/she runs the algorithm Encrypt(I_enc, K) of OABE to generate the ciphertext CT_K which is an encryption of the symmetric key with respect to I_enc. Finally, the data owner uploads the ciphertext (CT_F, CT_K) to S-CSP.

**New User Grant.** Assuming a user wants to join the system, he/she needs to be issued a private key on his/her access structure (resp. attribute set)
\(I_{\text{key}}\) from AA who then runs the outsourced key-issuing protocol. In concrete, AA outsources the operation of key-issuing by running the algorithm of \(O\)-\text{KeyGen-PreProc}(I_{\text{key}}, MK)\) to obtain an outsourcing key \(OK\). Using \(OK\), KG-CSP runs \(O\)-\text{KeyGen-Outsource}(I_{\text{key}}, OK)\) to generate a private key component \(SK_1\). Finally, AA generates the other private key component \(SK_2\) and assigns \(SK = (SK_1, SK_2)\) to the user.

**File Access.** Suppose a user wants to access and retrieve files of his/her interests. He/She firstly downloads the ciphertext \((CT_F, CT_K)\). To decrypt the ciphertext while relieving the local computation overhead, the user runs the outsourced decryption protocol with D-CSP by sending \(CT_K\) and the private key component \(SK_1\). If the user’s \(I_{\text{key}}\) in \(SK_1\) matches \(I_{\text{enc}}\) embedded in \(CT_K\), D-CSP is able to successfully compute and return the partially decrypted ciphertext \(CT'_K\). Upon receiving \(CT'_K\), the user performs complete decryption to get the symmetric key \(K\), with which he/she decrypts and retrieves the file \(F\).

**User Revocation.** Whenever there is a user to be revoked, a public parameter update technique in [40] is utilized. Specifically, AA determines a minimal set of attributes according to the user’s \(I_{\text{key}}\) and updates the corresponding components in \(PK\) and \(MK\). Then, AA updates private keys \(SK = (SK'_1, SK_2)\) for all the “affected” users by running the outsourced key-issuing protocol with KG-CSP. Additionally, to update “affected” ciphertexts having been stored in S-CSP, a re-encrypting key is generated by AA to be sent to S-CSP. S-CSP uses such a key to update the “affected” ciphertexts with the latest version of \(PK\). Notice that the main computation at AA side is updating private keys for “affected” users. Utilizing the outsourced key-issuing protocol, such complexity is minimized.

### 5.2 Security Analysis

Recall that two security requirements should be guaranteed, that is, data confidentiality and collusion-resistance. Thus, we provide the analysis on the both folds.

- **Data Confidentiality.** Since the file is encrypted with a hybrid encryption as \((CT_F, CT_K)\), to get any information about \(F\), the adversary should decrypt \(CT_K\) to retrieve the symmetric key \(K\). However such a key is protected by OABE. Thus, data confidentiality can be reduced to the confidentiality security of OABE. Moreover, the privacy of OABE ciphertext on S-CSP against outside users without \(I_{\text{key}}\) can be trivially guaranteed because its security definition inherits that in traditional ABE. Another attack on data confidentiality is launched by KG-CSP. Such an attack is modeled as type-II adversary by introducing an oracle \(O_{OK}(\cdot)\) in the corresponding security game of OABE. Specifically, we allow such an adversary to ask for outsourcing keys \(OK\) for all the users, but require it cannot successfully distinguish ciphertexts.
– **Collusion-Resistance.** Curious users can collude with both D-CSP and S-CSP to launch attack, which is modeled as type-I adversary in the definition of OABE. More precisely, we define such type of collusion by introducing oracles $O_{SK_1}(\cdot)$ and $O_{SK}(\cdot)$.

Therefore, the security of the attribute-based access control system is reduced to that of underlying OABE. In another word, if OABE is secure against both type-I and type-II adversaries, the proposed system is secure. In the following sections, we take our focus on OABE and attempt to provide secure OABE construction for attribute-based access control system.

### 6 Basic OABE Construction with Outsourced Decryption

In this section, we will define the access structure used in this paper and propose a basic OABE construction with only outsourced decryption. The security analysis of the proposed construction is provided as well.

#### 6.1 Access Structure

**Definition 4 (Access Structure).** Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $A \subseteq 2^{\{P_1, \ldots, P_n\}}$ is monotone if $\forall B, C : B \subseteq C$ then $C \in A$. An access structure (resp. monotone access structure) is a collection (resp. monotone collection) $A$ of non-empty subsets of $\{P_1, \ldots, P_n\}$. The sets in $A$ are called authorized sets.

Furthermore, we could define the predicate $\gamma(\cdot, \cdot)$ as follows

\[
\gamma(\omega; A) = \begin{cases} 
1 & \text{if } \omega \in A \\
0 & \text{otherwise}
\end{cases}
\]

In this paper, the role of the party is taken by attributes. Thus, the access structure $A$ contains the authorized sets of attributes. Specifically, the access structure represented by tree can be supported in this paper.

Let $T$ be an access tree, in which each interior node is a threshold gate (i.e. AND gate or OR gate) while the leaves are associated with attributes. A user is able to decrypt a ciphertext with a given key if and only if there is an assignment of attributes from the private key to leaf nodes of the tree such that the tree is satisfied.

To facilitate working with the access tree, we define a few notations and functions as follows.

– $\text{num}_x$ is the number of children of an interior node $x$. In order to uniquely identify each child, an ordering in the children of every node is defined, that is, the children of node $x$ is numbered from 1 to $\text{num}_x$. Therefore, if assuming $y$ is the child of node $x$, we could denote $\text{index}(y)$ as such number associated with the node $y$. 


– $k_x$ is the threshold value of an interior node $x$, specifically, when $k_x = 1$, the threshold gate at $x$ is OR gate and when $k_x = \text{num}_x$, that is an AND gate. We note that if $x$ is a leaf node it is described by an attribute and a threshold value $k_x = 1$.
– The function $\text{parent}(x)$ returns the parent of the node $x$ in the tree. $\text{attr}(x)$ returns the attribute associated with the leaf node $x$.

6.2 Proposed Construction

We only consider to outsource the decryption computation of ABE and propose a basic OABE construction with outsourced decryption. In another word, the KG-CSP will not be involved. For simplicity, this basic construction only considers to support for access structure described as $\mathcal{A} = \{\omega \subseteq \mathcal{U} : |\omega \cap \omega^*| \geq d\}$ where $\mathcal{U}$ is the attribute universe, $\omega$ and $\omega^*$ are attribute sets and $d$ is a predefined threshold value. Actually, it can be easily extended to an OABE supporting access structure represented by tree as shown in Section 7.2.

Before providing the construction, we define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set $S$ of elements in $\mathbb{Z}_p$ as $\Delta_{i,S} = \prod_{j \in S, j \neq i} \frac{x - j}{x - i}$.

Setup Phase.
– $\text{Setup}(\lambda)$: Define a bilinear group $\mathbb{G}$ of prime order $p$ with a generator $g$ and a bilinear map $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Next, define the attributes in universe $\mathcal{U}$ as elements in $\mathbb{Z}_p$. For simplicity let $n = |\mathcal{U}|$ and the first $n$ elements in $\mathbb{Z}_p$ (i.e. $1, 2, \ldots, n \mod p$) can be taken to be the universe. Select $x \in \mathbb{Z}_p$ and set $g_1 = g^x$. Pick $g_2, h, h_1, \ldots, h_n \in \mathbb{G}$. Finally, output the public parameter $PK = (g, g_1, g_2, h, h_1, \ldots, h_n)$ and the master key $MK = x$ which is kept secret by AA.

Key-Issuing Phase. As shown in Fig. 2, a hybrid policy $\mathcal{P} = \mathcal{P}_\theta \land \mathcal{P}_\omega$ is utilized in the key-issuing phase, where $\land$ is an AND gate connecting two sub-policies $\mathcal{P}_\omega$ and $\mathcal{P}_\theta$. More precisely, a default attribute $\theta$ is appended with each user’s attribute set and the master key $x$ is randomly split into $x_1$ and $x_2$ for each user to generate private key components on $\mathcal{P}_\omega$ and $\mathcal{P}_\theta$ respectively.

![Fig. 2. Illustration of Private Key Generation](image)

The key generation algorithm is described as follows.
KeyGen($\omega, MK$) : Upon receiving a private key request on attribute set $\omega$, the authority selects $x_1 \in R Z_p$ and sets $x_2 = x - x_1 \mod p$. Furthermore, select a $(d - 1)$-degree polynomial $q(\cdot)$ such that $q(0) = x_1$. Then, for each $i \in \omega$, choose $r_i \in R Z_p$, and compute $d_{i0} = g_2^{q(i)}(g_1 h_i)^{r_i}$ and $d_{i1} = g^{r_i}$. For the default attribute $\theta$, compute $d_{00} = g_2^{q(\theta)}(g_1 h)^{r_{\theta}}$ and $d_{01} = g^{r_{\theta}}$ by choosing $r_{\theta} \in R Z_p$. Finally, output the private key $SK = (SK_1, SK_2)$ where $SK_1 = \{d_{i0}, d_{i1}\}_{i \in \omega}$ and $SK_2 = \{d_{00}, d_{01}\}$.

Encryption Phase. Based on the logical split of user’s attribute private key, the default attribute $\theta$ should be embedded in each ciphertext to make the decryption successful. The encryption algorithm works as follows.

- **Encrypt($M, \omega'$) :** To encrypt a message $M$ with respect to an attribute set $\omega'$, select $s \in R Z_p$ and compute $C_0 = Me(g_1, g_2)^s$, $C_1 = g^s$, $E_0 = (g_1 h)^s$ and $E_i = (g_1 h_i)^s$ for each $i \in \omega'$. Finally, output the ciphertext $CT = (\omega' \cup \{\theta\}, C_0, C_1, \{E_i\}_{i \in \omega' \cup \{\theta\}})$.

Outsourced Decryption Phase. The outsourced decryption is consisted of two algorithms which are provided as follows.

- **O-Decrypt-Outsource($CT, SK_1$) :** Suppose that a ciphertext $CT$ is encrypted with an attribute set $\omega'$. After receiving the private key component $SK_1$ for attribute set $\omega$ sent from a user, D-CSP computes the partially decrypted ciphertext $CT'$ with $SK_1$ as shown in Fig. 3, where $\gamma_d(\omega, \omega') = 1$.
- **O-Decrypt-Dec($CT', SK$) :** Upon receiving $CT'$ from D-CSP, the user completely decrypts the ciphertext and gets a message $M$ as shown in Fig. 3.

Remark. Though D-CSP works as a “worker” in the outsourced decryption protocol described above, it enables another scenario in which D-CSP works as a “proxy”. More precisely, user can deliver and store $SK_1$ on D-CSP. In this way, D-CSP can automatically retrieve ciphertexts that the user is interested in and forward to him/her partially decrypted ones. For example, D-CSP could be users’ mail server, or the cloud storage server and D-CSP could be the same entity.
6.3 Security Analysis

The main challenge in our construction is to prevent against attacks from the collusion between users and D-CSP. However, such collusion is resistant due to the random split on master key $x$ for each user. More precisely, if two different users call for their private keys, AA will choose two random splits $(x_1, x_2)$ and $(x_1', x_2')$ such that $x_1 + x_2 = x \mod p$ and $x_1' + x_2' = x \mod p$. Note that $x_1$ and $x_1'$ are used to generate the private key component $SK_1$ and $SK_1'$ respectively, while $SK_2$ and $SK_2'$ are separately generated from $x_2$ and $x_2'$. In this sense, the ciphertext can be correctly decrypted only when $SK_1$ matches $SK_2$. Therefore, even if a group of curious users collude with D-CSP to obtain all $SK_1$, they cannot forge a valid private key for themselves to perform decryption successfully out of their scopes.

Since basic outsourced decryption is supported, we only need to consider the security against type-I adversary. Then, we have the following security result.

**Theorem 1.** The basic OABE scheme with outsourced decryption is secure against chosen-plaintext attack in selective model under DBDH assumption.

**Proof.** Please refer to Appendix.

7 Improved OABE Construction for Efficient Access Control System

In this section, based on the basic OABE, we further propose three OABE constructions supporting outsourced key-issuing. Utilizing these primitives, we can build an access control system with efficient key management.

7.1 OABE Construction for Efficient Access Control

Notice that in our basic construction, any adversary possessing either $SK_1$ or $SK_2$ cannot extract any useful information from the ciphertext. Thus, we
are able to outsource the operation of generating $SK_1$ to KG-CSP but remain computing $SK_2$ at AA. Considering on this, we propose an OABE construction with outsourced key-issuing and decryption. Since the other phases are identical to our basic construction, we only provide the outsourced key-issuing protocol as follows.

- **O-KeyGen-PreProc($\omega, MK$)**: The preprocessing algorithm in outsourced key-issuing protocol is run by AA. It picks $x_1 \in R Z_q$ and sets $x_2 = x - x_1 \mod q$. Finally, output $(OK, AK)$ where $OK = x_1$ and $AK = x_2$.

- **O-KeyGen-Outsource($\omega, OK$)**: The outsourcing algorithm is run by KG-CSP. It randomly selects a $(d - 1)$-degree polynomial $q(\cdot)$ with $q(0) = x_1$, and outputs $SK_1$ as shown in Fig. 5.

- **O-KeyGen-PostProc($SK_1, AK$)**: The postprocessing algorithm is run by AA. It computes and outputs $SK = (SK_1, SK_2)$ as shown in Fig. 5.

We have shown that the proposed construction is resistant to the type-I adversary in Section 6.3. Therefore, it is only needed to prove its security under the attack launched by the type-II adversary. Intuitively, in order to decrypt ciphertext, the adversary has to recover $e(g_1, g_2)^s$. The adversary could utilize Lagrange interpolation on $SK_1$ and $C_1 = g^s$ from ciphertext to recover the desired value. This will result in $e(g, g_2)^{sx_1}$ but blinded by $e(g, g_2)^{sx_2}$ which cannot be removed unless the other component of private key $SK_2$ is used.

Thus, we can get the following security result based on the analysis above.

**Theorem 2.** The proposed OABE construction with outsourced key-issuing and decryption is secure against chosen-plaintext attack launched by type-II adversary.

Combining Theorem and Theorem together, the proposed construction is secure against the chosen-plaintext attack.

### 7.2 OABE Construction for Fine-Grained Access Control

Though we describe our outsourcing key-issuing technique in the threshold ABE, it can be easily extended to be applied to the access tree-based KP-ABE scheme to enable fine-grained access control.

The idea behind this extension is to build a hybrid tree $T$ as shown in Fig. 6, where $\land$ and $\lor$ denote AND and OR gates respectively, and $A_i$ denotes the attribute. In this case, KG-CSP is to compute $SK_1$ with the tree-based access policy instead of threshold policy.

Suppose the access tree specified by user is denoted as $T_U$. Assuming the parameters have been assigned as the setup algorithm in Section 6.2, we provide the outsourced key-issuing protocol for access tree-based KP-ABE scheme as follows.

- **O-KeyGen-PreProc($T_U, MK$)**: Randomly pick a one-degree polynomial $q_R(\cdot)$ with $q_R(0) = x$. Set $x_1 = q_R(1)$ and $x_2 = q_R(2)$. Finally output $OK = x_1$ and $AK = x_2$. 
Fig. 5. Hybrid Tree Policy

- **O-KeyGen-Outsource**($\mathcal{T}_U, OK$) : Firstly, choose a $(k_x - 1)$-degree polynomial $q_x(\cdot)$ for each node $x$ (including leaves) in the tree $\mathcal{T}_U$ in a top-down manner. We note that the polynomial $q_x(\cdot)$ is chosen with the restriction that $q_x(0) = x_1$ if $x$ is the root node in $\mathcal{T}_U$, otherwise $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$. Let $Y_U$ be the set of leaf nodes in $\mathcal{T}_U$, then the private key component $SK_1$ is set to be $(\{g_{x}^{q_x(0)}(g_1 h_{\text{attr}(y)})^{r_y}, g^{s_y}\}_{y \in Y_U})$ where $r_y \in_R \mathbb{Z}_p$.

- **O-KeyGen-PostProc**($AK, SK_1$) : After generating the private key component $SK_2 = (\{g_{x}^{q_x(0)}(g_1 h_{\text{attr}(y)})^{r_y}, g^{s_y}\}_{y \in Y_U})$ where $r_\theta \in_R \mathbb{Z}_p$, AA outputs the private key $SK = (SK_1, SK_2)$.

### 7.3 OABE Construction with Reduced Trust on KG-CSP

In this section, we propose an improved OABE construction under a stronger adversary model namely RDoC (referred delegation of computation) model, in which the trust on KG-CSP is reduced. In RDoC model, two KG-CSPs are involved and at least one of them is honest. Actually, such model has been utilized by Hohenberger and Lysyanskaya [22] to securely outsource cryptographic computations and was later formulated in [9][12].

For simplicity, threshold policy is considered. Since the setup, encryption and outsourced decryption phases operate exactly as before, we only describe the outsourced key-issuing protocol with two KG-CSPs as follows.

- **O-KeyGen-PreProc**($\omega, MK$) : The algorithm is presented similar to that in our proposed construction in Section 7.1. The only difference is that AA does not directly send $x_1$ to KG-CSP, but makes a further random split to obtain $x_{11}$ and $x_{12}$ with $x_1 = x_{11} + x_{12} \mod p$. Finally, send $OK[j] = x_{1j}$ to KG-CSP$[j]$ for $j = 1, 2$.

- **O-KeyGen-Outsource**$[j](\omega, OK[j])$ : The KG-CSP$[j]$ randomly picks a $(d - 1)$-degree polynomial $q(\cdot)$ with $q(0) = x_{1j}$ and computes and returns $SK_1[j]$ as shown in Fig. 7.
- O-KeyGen-PostProc($SK_1[1], SK_1[2], AK$) : AA computes $SK_2$ and obtains $SK_1$ from $SK_1[1]$ and $SK_1[2]$ as shown in Fig. 6. Finally output $SK = (SK_1, SK_2)$.

**Fig. 6.** Outsourced Key-Issuing Protocol with Two KG-CSPs

### 8 Performance Evaluation

As shown in Fig. 7, we provide a thorough experimental evaluation of the construction proposed in Section 7.1. Our experiment is simulated with the pairing-based cryptography (PBC) library[31] on a Linux machine with Intel Core 2 processor running at 2.40 GHz and 2G memory.

Our analysis is in terms of four phases in the construction. The computation cost is constant at AA and user sides during outsourced key-issuing and decryption respectively. We also analyze the efficiency during both setup and encryption phases.

### 9 Related Work

**Attribute-based Encryption.** The notion of ABE, which was introduced as fuzzy identity-based encryption in [35], was firstly dealt with by Goyal et al. [20]. Two different and complementary notions of ABE were defined as KP-ABE and CP-ABE. A construction of KP-ABE was provided in the same paper [20], while the first CP-ABE construction supporting tree-based access structure in generic group model is presented by Bethencourt et al. [6].

Subsequently, a number of variants of ABE schemes have been proposed since its introduction. They range from extending its functionality to proposing schemes with stronger security proofs. Such as ABE schemes supporting for any kinds of access structures [13] [28] [33], ABE with multi-authorities [10] [11], accountable ABE [29], full secure ABE [25] [38] [26], decentralized ABE [27], etc.
Recently, a novel paradigm for ABE was provided [21][30]. In [21], Green et al. considered to outsource the decryption of ABE to eliminate the overhead at user side, while an outsourced ABE with outsourced encryption and decryption was presented in [42][30]. We point out that the outsourcing decryption technique in [21][42] is to blind user’s attribute private key by running a number of exponentiations. But such key blinded operation is eliminated in our construction in Section 6.2 through introducing a default attribute (actually, our technique provides a feasible way to realize the “piecewise key generation” property recently introduced in [34]). Moreover, the previous work all lacks of the consideration on the reducing overhead computation at attribute authority. Thirdly, in this paper we also consider a type of collusion attack in the sense that KG-CSP is dishonest and its defense strategy.

**Outsourcing Computation** To reduce the load at local, it always desires to deliver expensive computational tasks outside. Actually, the problem that how to securely outsource different kinds of expensive computations has drew much attention from theoretical computer science community [3][5][1][37]. But they are not suitable for reliving ABE computational overhead at user or authority side. To achieve this goal, the traditional approach is to utilize server-aided techniques [7][23][22]. However, previous work is oriented to accelerating the speed of exponentiation using untrusted servers. Directly utilizing these techniques in ABE will not work efficiently. Another approach might be to leverage recent gen-
eral outsourcing technique or delegating computation \[16\] based on fully homomorphic encryption or interactive proof system. However, Gentry \[17\] has shown that even for weak security parameters on “bootstrapping” operation of the homomorphic encryption, it would take at least 30 seconds on a high performance machine. Therefore, even if the privacy of the input and output can be preserved by utilizing these general techniques, the computational overhead is still huge and impractical.

10 Conclusion

In this paper, we propose an efficient attribute-based access control system in cloud computing. In our system, two CSPs namely KG-CSP and D-CSP are introduced as employees to finish the outsourced heavy tasks for user management and file access respectively. The overhead at both users and attribute authority sides is thus being minimized. A challenging issue in the proposed system is how to outsource the computational task to CSPs without any private information leakage. To deal with this issue, we formulate an underlying primitive namely OABE and provide several OABE constructions with outsourced key-issuing and decryption. Finally, through extensive experiments, it demonstrates that our OABE construction achieves efficient key-issuing and decryption at AA and user sides respectively.

References


Appendix A: Definition of Security Games

Our security definition is similar to [21], we elaborate the games in detail as follows.

RCCA Security Game for Type-I Adversary

**Setup.** Challenger runs Setup(λ) and gives PK to adversary.

**Phase 1.** Challenger initializes an empty table $T$ and set $D$ to provide adversary the oracles below.

- $\mathcal{O}_{SK_1}(I_{\text{key}})$. If there exists $(I_{\text{key}}, SK, SK_1)$ in $T$, return $SK_1$. Otherwise run the outsourced key-issuing protocol for $I_{\text{key}}$ to return $SK_1$ after adding $(I_{\text{key}}, SK, SK_1)$ into $T$.

- $\mathcal{O}_{SK}(I_{\text{key}})$. Set $D = D \cup \{I_{\text{key}}\}$. If there exists $(I_{\text{key}}, SK, SK_1)$ in $T$, return $SK$. Otherwise run the outsourced key-issuing protocol for $I_{\text{key}}$ to return $SK$ after adding $(I_{\text{key}}, SK, SK_1)$ into $T$.

- $\mathcal{O}_{M}(I_{\text{key}}, CT)$. If there exists $(I_{\text{key}}, SK, SK_1)$ in $T$, decrypt $CT$ and return $M$. Otherwise run the outsourced key-issuing protocol and add $(I_{\text{key}}, SK, SK_1)$ into $T$. Finally decrypt $CT$ and return $M$.

**Challenge.** Adversary submits two messages $M_0$ and $M_1$ as well as $I_{\text{enc}}^*$ satisfying $\gamma(I_{\text{key}}, I_{\text{enc}}^*) = 0$ for all $I_{\text{key}} \in D$. Challenger picks $b_1 \in_R \{0, 1\}$, encrypts $M_{b_1}$ under $I_{\text{enc}}^*$ and returns the resulting ciphertext $CT^*$.

**Phase 2.** Phase 1 is repeated with the restrictions: i) the queries $\mathcal{O}_{SK}(I_{\text{key}})$ where $\gamma(I_{\text{key}}, I_{\text{enc}}^*) = 1$ are not allowed; ii) the queries $\mathcal{O}_M(\cdot, \cdot)$ resulting in $M_0$ or $M_1$ will be replaced by a special message.

**Guess.** Adversary outputs a guess $b_1'$ of $b_1$.

RCCA Security Game for Type-II Adversary

**Setup.** It is identical to the setup phase in the RCCA security game for type-I adversary.

**Phase 1.** Challenger initializes an empty table $T$ to provide adversary the oracles below.
If there exists $(I_{key}, SK, OK)$ in $T$, return $OK$. Otherwise run the outsourced key-issuing protocol for $I_{key}$ to return $OK$ (an intermediate output) after adding $(I_{key}, SK, OK)$ into $T$.

- $O_M(I_{key}, CT)$. It is identical to $O_M(I_{key}, CT)$ in the RCCA security game for type-I adversary except that the entry to be added is in the form of $(I_{key}, SK, OK)$.

**Challenge.** Adversary submits two messages $M_0$ and $M_1$ as well as $I^*_e$. Challenger picks $b_1 \in \{0,1\}$ and encrypts $M_{b_1}$ under $I^*_{enc}$. Finally return the resulting ciphertext $CT^*$.

**Phase 2.** Phase 1 is repeated with the restriction that the queries $O_M(\cdot, \cdot)$ resulting in $M_0$ or $M_1$ will be replaced by a special message.

**Guess.** Adversary outputs a guess $b'_1$ of $b_1$.

Denote $A_i$ as type-$i$ adversary for $i = I, II$. Then, their advantages in attacking the OABE scheme $E$ is measured by the probability $Adv^\text{RCCA}_{E, A_i}(\lambda) = |Pr[b_i = b'_1] - \frac{1}{2}|$ respectively.

**Appendix B: Proof of Theorem 1**

**Proof.** Assume there exists an adversary $A$ breaks the proposed scheme, we can build a simulator $S$ that uses $A$ as a sub-algorithm to solve the DBDH problem as follows.

The challenger firstly flips a fair binary coin $\mu$ outside of $S$’s view. If $\mu = 0$, $S$ is given $(X = g^x, Y = g^y, Z = g^z, T = e(g, g)^{xy})$; otherwise, $S$ is given $(X = g^x, Y = g^y, Z = g^z, T = e(g, g)^x)$ for $x, y, z \in R \mathbb{Z}_p$. $S$ is asked to output a value $\mu'$ as the guess for $\mu$. We provide the simulation as follows.

**Init.** The simulator $S$ runs $A$ and receives a challenge attribute set $\omega^*$ from $A$.

**Setup.** $S$ assigns the public parameter as follows. It sets $g_1 = X, g_2 = Y$ and $h = g_1^{-1} g^{-\alpha}$ where $\alpha \in R \mathbb{Z}_p$. For $i \in \omega^*$, it selects $\alpha_i \in R \mathbb{Z}_p$ and sets $h_i = g_i^{-1} g^{\alpha_i}$. For $i \notin \omega^*$, it selects $\alpha_i \in R \mathbb{Z}_p$ and sets $h_i = g^{\alpha_i}$. Finally, $S$ sends the public parameter $PK = (g, g_1, g_2, h, h_1, \ldots, h_n)$ to $A$, where $n$ is the number of attributes in universe.

**Phase 1.** $S$ initializes an empty table $T$ and $A$ is provided two types of oracles as follows.

- $O_{SK_1}(\omega)$: Upon receiving the private key component $SK_1$ request on $\omega$, $S$ checks whether the entry $(\omega, \cdot, SK_1)$ exists in $T$. If so return $SK_1$; otherwise simulate as follows.

  - If $|\omega \cap \omega^*| < d$, $S$ picks $x_2 \in R \mathbb{Z}_p$ and defines three sets $\Gamma, \Gamma'$ and $\Gamma''$, where $\Gamma = \omega \cup \omega^*, |\Gamma'| = d - 1, \Gamma' \subseteq \Gamma' \subseteq \omega$ and $S = \Gamma'' \cup \{0\}$. Then, for each $i \in \Gamma''$, compute $d_{i0} = g_2^i (g_1 h_i)^{\tau_i}$ and $d_{i1} = g^i$ where $\tau_i \in R \mathbb{Z}_p$. For each $i \in \omega \setminus \Gamma''$, set $r_i = \sigma_i^{\Delta_i} (g_1 h_i)^{r_i}$ by choosing $r_i \in R \mathbb{Z}_p$. Finally, compute $d_{i0} = g_2^i \Delta_i (g_1 h_i)^{r_i}$.

- If $|\omega \cap \omega^*| = d$, $S$ picks $x_2 \in R \mathbb{Z}_p$ and defines three sets $\Gamma, \Gamma'$ and $\Gamma''$, where $\Gamma = \omega \cup \omega^*, |\Gamma'| = d - 1, \Gamma' \subseteq \Gamma' \subseteq \omega$ and $S = \Gamma'' \cup \{0\}$. Then, for each $i \in \Gamma''$, compute $d_{i0} = g_2^i (g_1 h_i)^{\tau_i}$ and $d_{i1} = g^i$ where $\tau_i \in R \mathbb{Z}_p$. For each $i \in \omega \setminus \Gamma''$, set $r_i = \sigma_i^{\Delta_i} (g_1 h_i)^{r_i}$ by choosing $r_i \in R \mathbb{Z}_p$. Finally, compute $d_{i0} = g_2^i \Delta_i (g_1 h_i)^{r_i}$.
and $d_{1i} = g_2^{\Delta_{u,i}(i)}$. The intuition behind these assignments is that a random $d-1$ degree polynomial $q(\cdot)$ is implicitly determined by choosing $q(i) = r_i$ for $i \in T'$ and $q(0) = x - x_2$.

- Otherwise (i.e. $|\omega \cap \omega^*| \geq d$), $S$ picks $x_1 \in \mathbb{Z}_p$ and randomly selects a $(d-1)$-degree polynomial $q(\cdot)$ with $q(0) = x_1$. Then, for each attribute $i \in \omega$, $d_{0i} = g_2^{q(i)}(g_1 h_i)^{r_i}$ and $d_{1i} = g^{r_i}$ where $r_i \in \mathbb{Z}_p$.

Finally, after adding the entry $(\omega, *, SK_1)$ into $T$, $S$ returns $SK_1 = \{(d_{0i}, d_{1i}) \mid i \in \omega\}$.

$O_{SK}(\omega)$: Upon receiving a private key request on $\omega$ with $|\omega \cap \omega^*| < d$, $S$ checks whether the entry $(\omega, SK, *)$ exists in $T$. If so return $SK$; otherwise if the value $x_2$ for such entry has not been selected, $S$ picks $x_2 \in \mathbb{Z}_p$ and the remaining simulation is similar to the first case (i.e. $|\omega \cap \omega^*| < d$) in $O_{SK_1}(\cdot)$ to obtain $SK_1$, and compute $SK_2 = (d_{00} = g_2^x(g_1 h)^r, d_{01} = g^x)$, where $r_0 \in \mathbb{Z}_p$. Finally, after adding $(\omega, SK, SK_1)$ into $T$, $S$ returns $SK = (SK_1, SK_2)$.

**Challenge.** Two challenge messages $M_0$ and $M_1$ are chosen by $A$. The simulator $S$ flips a fair binary coin $\nu$ and generates the ciphertext of $M_\nu$ as $CT^* = (\omega^* \cup \{\theta\}, M_\nu, T, g^v, g^{-\omega}, \{g^{\omega_i}\}_{i \in \omega^*})$. Note that: i) If $\mu = 0$, then $T = e(g, g)^{xyz}$. Let $s = z$ and we have $C_0 = M_\nu T = M_\nu e(g, g)^{xyz} = M_\nu e(g_1, g_2)^z, C_1 = g^v, E_0 = g^{-\omega} = (g_1 g_1^{-1} g^{-\alpha})^z = (g_1 h)^z$ and $E_i = g^{\omega_i} = (g_1 g_1^{-1} g^{\alpha i})^z = (g_1 h_i)^z$ for $i \in \omega^*$. Therefore, the ciphertext is a random encryption of the message $M_\nu$ under the attribute set $\omega^*$; ii) if $\mu = 1$, then $T = e(g, g)^{v}$. We then have $C_0 = M_\nu e(g, g)^v$. Since $v$ is random, $C_0$ will be a random element in $G_T$ from $A$’s view and ciphertext contains no information about $M_\nu$.

**Phase 2.** Phase 1 is repeated with the restriction that $A$ cannot issue query $O_{SK}(\omega)$ with $\gamma_d(\omega, \omega^*) = 1$.

**Guess.** $A$ outputs a guess $\nu'$ of $\nu$. If $\nu' = \nu$, $S$ outputs $\mu' = 0$ to indicate that it was given a DBDH-tuple; otherwise, it outputs $\mu' = 1$ to indicate it was given a random 4-tuple.

In the case where $\mu = 1$, $A$ has no information about $\nu$. Therefore, we have $Pr[\nu' \neq \nu | \mu = 1] = \frac{1}{2}$. Since $S$ guesses $\mu' = 1$ when $\nu \neq \nu'$, we have $Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. Therefore, we have $Pr[\nu' | \mu = 0] = \frac{1}{2} + \epsilon$. Since $S$ outputs $\mu' = 0$ when $\nu = \nu'$, we have $Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$. The overall advantage of $S$ is $\frac{1}{2} Pr[\mu' = \mu | \mu = 0] + \frac{1}{2} Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} (\frac{1}{2} + \epsilon) + \frac{1}{2} - \frac{1}{2} - \frac{1}{2} - \frac{1}{2} = \frac{1}{2} \epsilon$. 
