available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers

G | Wa [a,*], Q L | [a,b], J W | [b], M . G [c]

[a] School of Information Science and Engineering, Central South University, Changsha, Hunan Province 410083, PR China
[b] Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
[c] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200030, PR China

ARTICLE INFO

Article history:
Received 27 December 2010
Received in revised form
25 April 2011
Accepted 14 May 2011

Keywords:
Cloud computing
Hierarchical attribute-based encryption
Fine-grained access control
User revocation

ABSTRACT

With rapid development of cloud computing, more and more enterprises will outsource their sensitive data for sharing in a cloud. To keep the shared data confidential against untrusted cloud service providers (CSPs), a natural way is to store only the encrypted data in a cloud. The key problems of this approach include establishing access control for the encrypted data, and revoking the access rights from users when they are no longer authorized to access the encrypted data. This paper aims to solve both problems. First, we propose a hierarchical attribute-based encryption scheme (HABE) by combining a hierarchical identity-based encryption (HIBE) system and a ciphertext-policy attribute-based encryption (CP-ABE) system, so as to provide not only fine-grained access control, but also full delegation and high performance. Then, we propose a scalable revocation scheme by applying proxy re-encryption (PRE) and lazy re-encryption (LRE) to the HABE scheme, so as to efficiently revoke access rights from users.

1. Introduction

Cloud computing, as an emerging computing paradigm, enables users to remotely store their data in a cloud, so as to enjoy services on-demand. Migrating data from the user side to the cloud offers great convenience to users, since they can access data in the cloud anytime and anywhere, using any device, without caring about the capital investment to deploy the hardware infrastructures. Especially for small and medium-sized enterprises with limited budgets, they can achieve cost savings and the flexibility to scale (or shrink) investments on-demand, by using cloud-based services to manage projects, enterprise-wide contacts and schedules, and the like.

However, allowing a cloud service provider (CSP), operated for making a profit, to take care of confidential corporate data, raises underlying security and privacy issues. For instance, an untrustworthy CSP may sell the confidential information about an enterprise to its closest business competitors for making a profit. Therefore, a natural way to keep sensitive data confidential against an untrusted CSP is to store only the encrypted data in the cloud.

We consider the following application scenario (see Fig. 1): Company A pays a CSP for sharing corporate data in cloud servers. Suppose the sales department (SD), the research and development department (RDD), and the finance department (FD) are collaborating in Project X. The SD manager wants to store an encrypted user requirement analysis (URA) in the cloud, so that only the personnel that have certain certificates can access the document. For instance, the SD manager may specify an access control policy for this URA, as shown in Fig. 2.

* *Corresponding author.*
E-mail address: csgjwang@mail.csu.edu.cn (G. Wang).
URL: http://trust.csu.edu.cn/faculty/csgjwang
0167-4048/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2011.05.006

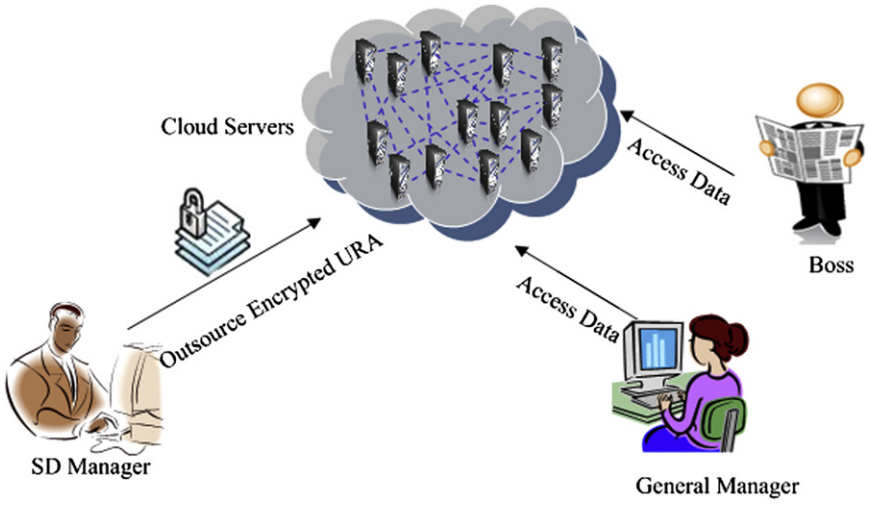

Fig. 1 – Sample application scenario.

In Fig. 2, the access control policy can be expressed as a Boolean formula over attributes. Each attribute consists of a web site specifying which party administers the attribute and an identifier describing the attribute itself, both of which can be represented as strings and concatenated with a single colon character as a separator. The slash "/" in each web site denotes a concatenation between the superior and the subordinate.

The intuition behind this access control policy is that this URA should only be accessed by the boss and the general manager of the enterprise, the members of Project X, and all the department managers who are involved in Project X. Furthermore, the party that administers attributes "isBoss", "isGeneralManager", and "inProjectX" is superior to the party that administers attributes "isDepartmentManager", "inSD", "inRDD", and "inFD".

In the above application scenario, the encrypter does not know the exact identities of the intended recipients, but rather he only has a way to describe them using certain descriptive attributes. Therefore, the adopted encryption system should support an attribute-based access structure.

Flexible encryption schemes such as ciphertext-policy attribute-based encryption (CP-ABE), can be adopted to provide a fine grain access control for the encrypted data. CP-ABE allows to encrypt data specifying an access control policy over attributes, so that only users with a set of attributes satisfying this policy can decrypt the corresponding data. For example, the data encrypted using the access structure $(a_1 \wedge a_2) \vee a_3$ means that only the user with attributes a_1 and a_2, or the user with attribute a_3, can decrypt the data.

However, since the data is outsourced to the cloud, the adopted CP-ABE scheme should also provide the following properties:

(1) **High performance**. In the cloud-computing environment, users may access data anytime and anywhere using any device. When a user wants to access data using a thin client with limited bandwidth, CPU, and memory capabilities, the CP-ABE scheme should be of high performance. That is, the communication costs and computation costs introduced by the CP-ABE scheme should be low enough, so that the user can successfully retrieve data from the cloud, and then decrypt it using the thin client.
(2) **Full delegation**. In a large-scale enterprise with many employees, each employee needs to request secret keys from the attribute authority (AA), when he joins the enterprise. If all these employees require their secret keys from one AA, there will be a performance bottleneck on the AA. To reduce the workload on the AA, some CP-ABE schemes provide key delegation between users, which enables a user to generate attribute secret keys containing a subset of his own attribute secret keys for other users. However, a full delegation mechanism, which can embody the hierarchical structure in the enterprises, is more applicable to the environment of enterprises outsourcing data in a cloud. Full delegation means key delegation between AAs, where each AA independently makes decisions on the structure and semantics of its attributes.

```
http://www.companyA.com:  isBoss                                  OR
http://www.companyA.com:  isGeneralManager                        OR
http://www.companyA.com:  inProjectX                              OR
  (http://www.companyA.com/Department: isDepartmentManager  AND
      ( http://www.companyA.com/Department: inSD     OR
        http://www.companyA.com/Department: inRDD    OR
         http://www.companyA.com/Department: inFD     ) )
```

Fig. 2 – Sample access control policy of URA.

(3) **Scalable revocation.** In the case of a large-scale enterprise with a high turnover rate, a scalable revocation scheme is a must. That is, the enterprise can revoke data access rights from users once they are no longer its employees. A user whose permission is revoked will still retain the keys issued earlier, and thus can still decrypt data in the cloud. The traditional revocation scheme usually requires the AAs to periodically re-encrypt data, and re-generate new secret keys to remaining authorized users. This approach will cause heavy workload on the AAs. A more scalable approach is to take advantage of the abundant resources in a cloud by allowing the AAs to delegate the CSP to re-encrypt data and re-generate keys to users, under the environment that the CSP knows nothing about the data and keys.

Based on the above-mentioned analysis, it is needed to propose a secure data-sharing scheme, which simultaneously achieves high performance, full delegation, and scalable revocation. Our contributions are as follows:

1. We propose a hierarchical attribute-based encryption (HABE) model, by combining the hierarchical identity-based encryption (HIBE) system and the CP-ABE system. The HABE model, which incorporates the property of hierarchical generation of keys in the HIBE system, and the property of flexible access control in the CP-ABE system, is more applicable to the environment of enterprises sharing data in the cloud.
2. We propose a HABE scheme based on the proposed model, which requires only a constant number of bilinear map operations during decryption, to provide high performance.
3. We propose a scalable revocation scheme, which allows to delegate most of computation intensive tasks in revocation to the CSPs without disclosing data contents, by applying proxy re-encryption (PRE) and lazy re-encryption (LRE) to the HABE scheme.

1.1. O a a

The rest of this paper is organized as follows: We begin with a discussion of related work in Section 2, and present some preliminaries in Section 3. Then, we provide an overview and an efficient construction for the HABE scheme, in Sections 4 and 5, respectively. Next, we analyze the performance and security for the HABE scheme in Section 6, and outline a scalable revocation scheme in Section 7. Then, we describe how the HABE scheme and the revocation scheme work at the system level in Section 8. Finally, we conclude this paper in Section 9.

2. Related work

2.1. H a c ca -ba c

In an identity-based encryption (IBE) system (Boneh and Franklin, 2001), there is only one private key generator (PKG) to distribute private keys to each user, which is undesirable for a large network because PKG has a burdensome job. Gentry and Silverberg (2002), who have been dedicated to reducing the workload on the root PKG, introduced a HIBE scheme. Their scheme with total collusion resistance at an arbitrary number of levels, has chosen ciphertext security under the random oracle model and the Bilinear Diffie–Hellman (BDH) assumption. A subsequent construction by Boneh and Boyen (2004) proposed a HIBE system with selective-ID security under the BDH assumption without random oracles. In both constructions, the length of ciphertext and private keys, as well as the time during encryption and decryption, grows linearly with the depth of a recipient in the hierarchy. For better performance, Boneh et al. (2005) proposed an efficient HIBE system which requires only a constant length of ciphertext and a constant number of bilinear map operations during decryption. In recent work, Gentry and Halevi (2009) proposed a fully secure HIBE scheme by using identity-based broadcast encryption with key randomization, and Waters (2009) achieved full security in systems under a simple assumption by using a dual system encryption.

2.2. A bl -ba c

Sahai and Waters (2005) introduced the notion of attribute-based encryption (ABE). Based on their work, Goyal et al. (2006) proposed a fine-grained access control ABE scheme, which supports any monotonic access formula. Their scheme is characterized as key-policy ABE (KP-ABE) since the access structure is specified in the private key, while the attributes are used to describe the ciphertext. A subsequent construction by Ostrovsky et al. (2007) allows for non-monotonic access structures.

Bethencourt et al. (2007) introduced a ciphertext-policy ABE (CP-ABE) scheme, in which the roles of the ciphertext and keys are reversed in contrast with the KP-ABE scheme. Muller et al. (2008) constructed an efficient distributed attribute-based encryption (DABE) scheme that requires a constant number of bilinear map operations during decryption, using disjunctive normal form (DNF) policy. Both of the above mentioned CP-ABE schemes provide a proof of selective security under the generic bilinear group model and the random oracle model. In our previous work, a conjunctive fuzzy and precise identity-based encryption (FPIBE) (Wang et al.) scheme is proposed for secure data sharing in cloud servers. The FPIBE scheme is able to efficiently achieve a flexible access control by separating the access control policy into two parts: a recipient identity (ID) set and an attribute-based access control policy. Using the FPIBE scheme, a user can encrypt data by specifying a recipient ID set, or an access control policy over attributes, so that only the user whose ID belonging to the ID set or attributes satisfying the access control policy can decrypt the corresponding data. However, it does not address the scalability issue.

In recent work, to achieve a scalable revocation mechanism in cloud computing, Yu et al. (2010b) combined KP-ABE, proxy re-encryption (PRE) (Blaze et al., 1998), and lazy re-encryption (LRE) (Kallahalla et al., 2003). However, the technique in Yu et al (Boneh et al., 2005) cannot be applied directly to combine PRE and CP-ABE. Since in contrast to KP-ABE, the access structure is associated with data other than the user attribute key. The first combination of CP-ABE and PRE technique was first introduced by our previous work (Wang et al., 2010) and Yu et al. (2010a). The insufficiencies in the two

schemes are as follows: The former lacks security proof for the encrypted scheme and systematical descriptions for the revocation scheme, and the latter constructs a CP-ABE supporting only "AND" semantic in the access control and does not support key delegation.

2.3. S a

The characteristics of a HIBE system and a CP-ABE system are supporting "full delegation" and "fine-grained access control over attributes", respectively. Therefore, a natural way is to combine these encryption models. This is a non-trivial task, since the former is designed for encrypting to an exact recipient, however, the latter is designed for encrypting to a set of attributes. Furthermore, we found that the HIBE scheme in Gentry and Silverberg (2002) supports "one-to-many" encryption: An encrypted file can be decrypted by a recipient and all his ancestors, using their own secret keys, respectively, which can be regarded as a meeting point with a CP-ABE system. Therefore, we construct public/secret keys as Gentry and Silverberg (2002), which are the intuitions of the "one-to-many" property. Then, inspired by Muller et al. (2008), we found that an encryption scheme achieves not only better performance, but also the combination of a HIBE system and a CP-ABE system, using the DNF access control policy. Finally, inspired by Yu et al. (2010b), we also apply PRE and LRE in our scheme to achieve a scalable revocation mechanism.

3. Preliminaries

3.1. S

We assume that the system is composed of the following parties: the CSP, the trusted third party (TTP), enterprise users, end users, and internal trusted parties (ITPs). The first two parties can be easily understood: the CSP operates a large number of interconnected cloud servers with abundant storage capacity and computation power to provide high-quality services, and the TTP is responsible for generating keys for the CSP and enterprise users. We use Fig. 3 as an example to illustrate the last three parties: Company A that pays for sharing corporate data in cloud servers is an enterprise user, all personnel in the company, who share data in cloud servers are end users, and a department in the company that delegates keys inside the company is the ITP.

3.2. S d

As described in Haclgiimfi et al. (2002), there are two main attacks under such a circumstance, i.e., external attacks initiated by unauthorized outsiders, and internal attacks initiated by an honest but curious CSP (Yu et al., 2010b), as well as malicious end users. Since data is stored with the encrypted form in the cloud and communication channels between users and cloud are assumed to be secured under existing security protocols such as SSL, we only consider the internal attacks. The honest but curious CSP will always correctly execute a given protocol, but may try to learn some additional

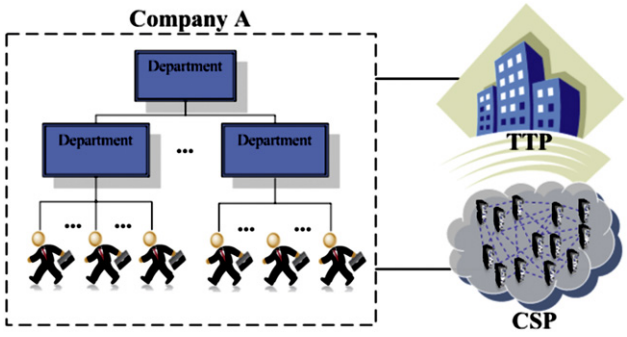

Fig. 3 – System model.

information about the stored data. The malicious end user wants to access the data that he is ineligible to decrypt. We assume that the CSP will not collude with the end users, since the CSP is considered to be honest but curious.

3.3. HABE

Corresponding to the system model, the HABE model (see Fig. 4), which integrates properties in both a HIBE model and a CP-ABE model, consists of a root master (RM) and multiple domains, where the RM functions as the TTP, and the domains are enterprise users. More precisely, a domain consists of many domain masters (DMs) corresponding to the ITPs and numerous users corresponding to end users.

The RM, whose role closely follows the root PKG in a HIBE system, is responsible for generation and distribution of system parameters and domain keys. The DM, whose role integrates both the properties of the domain PKG in a HIBE system and AA in a CP-ABE system, is responsible for delegating keys to the DMs at the next level and distributing secret keys to users. Specially, we enable the leftmost DM at the second level to administer all the users in a domain, just as the personnel office administers all personnel in an enterprise, and not to administer any attribute. Notice that other DMs administer an arbitrary number of disjoint attributes, and have a full control over the structure and semantics of their attributes.

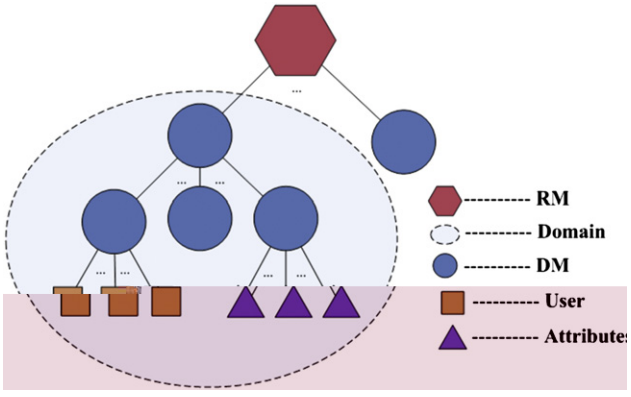

Fig. 4 – HABE model.

In the HABE model, we first mark each DM and attribute with a unique identifier (ID), but mark each user with both an ID and a set of descriptive attributes, where ID is an arbitrary string corresponding to some unique information about an entity. Then, as Gentry and Silverberg (2002), we enable each entity's secret key to be extracted from the DM administering itself. Each entity's public key, which denotes its position in the model, is an ID-tuple consisting of the public key of the DM administering itself and its own ID. For example, the public key of DM_i with ID_i is in the form of (PK_{i-1}, ID_i), the public key of user $\mathcal{U}$ with ID_u is in the form of $(PK_\blacklozenge, ID_u)$, and the public key of attribute a with ID_a is in the form of (PK_i, ID_a), where PK_{i-1}, $PK_\blacklozenge$, and PK_i are assumed to be the public keys of the DMs that administer DM_i, $\mathcal{U}$, and a, respectively.

4. Overview of the HABE scheme

4.1. Sc

Based on the HABE model, we propose a HABE scheme, in which the access control policies are expressed in DNF, and all attributes in one conjunctive clause should be administered by the same DM. For example, the access control policy in Fig. 2 can be transformed into DNF, as shown in Fig. 5.

In the HABE scheme, there are multiple keys with different usages. Therefore, we first provide a summary of the most relevant keys to serve as a quick reference (see Table 1). Then, we define the HABE scheme, by presenting randomized polynomial time algorithms as follows:

1. $Setup(K) \rightarrow (params, MK_0)$: The RM takes a sufficiently large security parameter K as input, and outputs system parameters *params* and root master key MK_0.
2. $CreateDM(params, MK_i, PK_{i+1}) \rightarrow (MK_{i+1})$: Whether the RM or the DM generates master keys for the DMs directly under it using *params* and its master key.
3. $CreateUser(params, MK_i, PK_u, PK_a) \rightarrow (SK_{i,u}, SK_{i,u,a})$: The DM first checks whether $\mathcal{U}$ is eligible for a, which is administered by itself. If so, it generates a user identity secret key and a user attribute secret key for $\mathcal{U}$, using *params* and its master key; Otherwise, it outputs "NULL".
4. $Encrypt(params, f, \mathbb{A}, \{PK_a | a \in \mathbb{A}\}) \rightarrow (CT)$: A user takes a file f, a DNF access control policy $\mathbb{A}$, and public keys of all attributes in $\mathbb{A}$, as inputs, and outputs a ciphertext CT.
5. $Decrypt(params, CT, SK_{i,u}, \{SK_{i,u,a} | a \in CC_j\}) \rightarrow (f)$: A user, whose attributes satisfy the j-th conjunctive clause CC_j, takes *params*, the ciphertext, the user identity secret key, and the user attribute secret keys on all attributes in CC_j, as inputs, to recover the plaintext.

4.2. S d

We define the semantical security (Boneh and Franklin, 2001) for the HABE scheme in terms of a game as follows:

4.2.1. Setup

The challenger runs the *Setup* algorithm when inputting a sufficiently large security parameter K, and gives *params* to adversary $\mathcal{A}$.

4.2.2. Phase 1

Adversary $\mathcal{A}$ can query any user key of his choice. When adversary $\mathcal{A}$ queries user $\mathcal{U}$'s user attribute secret key on attribute a, which is administered by DM_i, the challenger first runs the *CreateDM* algorithm to generate keys for DM_i, and then runs *CreateUser* algorithm to generate an identity secret key $SK_{i,u}$ and an attribute secret key $SK_{i,u,a}$, for $\mathcal{U}$. These queries may be asked adaptively.

4.2.3. Challenge

Once adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs a DNF access control policy $\mathbb{A}$ and two equal length plaintexts f_0, f_1 on which it wishes to be challenged. The only constraint is that adversary $\mathcal{A}$ enables none of the users to possess user attribute secret keys on all attributes in any of **A**'s conjunctive clauses in Phase 1. The challenger picks a random bit $b \in \{0,1\}$, sets $C_{f_b} = Encrypt(params, f_b, \mathbb{A}, \{PK_a | a \in \mathcal{A}\})$, and sends C_{f_b} as a challenge to adversary $\mathcal{A}$.

4.2.4. Phase 2

Adversary $\mathcal{A}$ issues more user attribute secret key queries with the same constraints as those in the challenge. The challenger responds as that in Phase 1.

4.2.5. Guess

$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, and wins the game if $b = b'$.

We define $\mathcal{A}$'s advantage in breaking the HABE scheme to be $Adv_{\mathcal{A}}(K) = |Pr[b = b'] - 1/2|$. The HABE scheme is semantically secure if for any polynomial time adversary $\mathcal{A}$, the function $Adv_{\mathcal{A}}(K)$ is negligible.

```
http://www.companyA.com:  isBoss                          OR
http://www.companyA.com:  isGeneralManager                OR
http://www.companyA.com:  inProjectX                      OR
    (http://www.companyA.com/Department:  isDepartmentManager     AND
     http://www.companyA.com/Department:  inSD )                  OR
    ( http://www.companyA.com/Department  isDepartmentManager     AND
      http://www.companyA.com/ Department:  inRDD  )               OR
    ( http://www.companyA.com/Department:  isDepartmentManager     AND
          http://www.companyA.com/Department:  inFD )
```

Fig. 5 – DNF access control policy of URA.

Table 1 – Summary of keys.

Key	Description	Usage
MK_0	Root master key	Creation of master key for DMs at the first level
PK_i	DM_i's public key	Creation of MK_i
MK_i	DM_i's secret key	Creation of user identity secret key and user attribute secret key
PK_u	$\mathcal{U}$'s public key	Creation of user identity secret key and user attribute secret key
$SK_{i,u}$	$\mathcal{U}$'s identity secret key requested from DM_i	Decryption
$SK_{i,u,a}$	$\mathcal{U}$'s attribute secret key requested from DM_i on attribute a	Decryption
PK_a	a 's public key	Creation of user attribute secret key

5. Our construction

In this section, we construct the HABE scheme using bilinear map (Boneh and Franklin, 2001). Let $\mathcal{IG}$ be a BDH parameter generator (Boneh and Franklin, 2001). We present the HABE scheme as follows:

1. Setup: The RM first picks $mk_0 \in \mathbb{Z}_q$, and then chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, three random oracles $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n, and $H_A : \{0,1\}^* \to \mathbb{Z}_q$, and a random generator $P_0 \in \mathbb{G}_1$. Let $Q_0 = mk_0 P_0 \in \mathbb{G}_1$. The system parameters $params = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, Q_0, H_1, H_2, H_A)$ will be publicly available, while $MK_0 = (mk_0)$ will be kept secret.
2. CreateDM: To generate the master key for DM_{i+1}, whose public key is PK_{i+1}, DM_{i+1}'s parent, whose master key is MK_i, first chooses a random oracle $H_{mk_{i+1}} : \{0,1\}^* \to \mathbb{Z}_q$, where $mk_{i+1} \in \mathbb{Z}_q$ is the index of the random oracle. Then, it computes $SK_{i+1} = SK_i + mk_i P_{i+1}$ where $P_{i+1} = H_1(PK_{i+1}) \in \mathbb{G}_1$, and $Q_{i+1} = mk_{i+1} P_0 \in \mathbb{G}_1$. Finally, it sets $MK_{i+1} = (mk_{i+1}, Hmk_{i+1}, SK_{i+1}, Q - tuple_{i+1})$, where $Q\text{-tuple}Q - \text{tuple}_{i+1} = (Q - \text{tuple}_i, Q_{i+1})$. Here, we assume that SK_0 is an identity element of $\mathbb{G}_1$, and $Q - \text{tuple}_0 = (Q_0)$.
3. CreateUser: When user $\mathcal{U}$ requests the user identity secret key, and the user attribute secret key on attribute a, DM_i first checks whether $\mathcal{U}$ is eligible for a, and a is administered by itself. If so, it first computes $mk_u = H_A(PK_u) \in \mathbb{Z}_q$, and then sets $SK_{i,u} = (Q - \text{tuple}_{i-1}, mk_i mk_u P_0)$, and $SK_{i,u,a} = SK_i + mk_i mk_u P_a \in \mathbb{G}_1$, where $P_a = H_{mk_i}(PK_a) P_0 \in \mathbb{G}_1$; Otherwise, it outputs "NULL".
4. Encrypt: Given a DNF access control policy $\mathbb{A} = \bigvee_{i=1}^{N} (CC_i) = \bigvee_{i=1}^{N} (\bigwedge_{j=1}^{n_i} a_{ij})$, where $N \in \mathbb{Z}^+$ is the number of conjunctive clauses in $\mathbb{A}$, $n_i \in \mathbb{Z}^+$ is the number of attributes in the i-th conjunctive clause CC_i, and a_{ij} is the j-th attribute in CC_i. Let DM it_i with (ID i_1,..., ID it_i) be the DM administering all attributes in CC_i, where ID_{ik} for $1 \le k < t_i$ are IDs of DM it_i's ancestors, and $ID_{i1} = ID_1$ is ID of the DM at the first level in a domain. The sender:
   (a) Picks a random element $r \in \mathbb{Z}_q$, sets $n_{\mathbb{A}}$ to be the lowest common multiple (LCM) of $n_1,\ldots,n_N$, and computes $U_0 = rP_0$, $U_{12} = rP_{12}$, ..., $U_{1t_1} = rP_{1t_1}$, $U_1 = r\sum_{j=1}^{n_1} P_{a_{1j}}$, ..., $U_{N2} = rP_{N2}$, ..., $U_{Nt_N} = rP_{Nt_N}$, $U_N = r\sum_{j=1}^{n_N} P_{a_{Nj}}$, and $V = f \oplus H_2(\hat{e}(Q_0, rn_{\mathbf{A}} P_1))$, where $P_{ij} = H_1(PK_{ij}) \in \mathbb{G}_1$ for $1 \le j \le t_i$ and $1 \le i \le N$, and $P_{a_{ij}} = H_{mk_{it_i}}(PK_{a_{ij}}) P_0 \in \mathbb{G}_1$ for $1 \le j \le n_i$ and $1 \le i \le N$.
   (b) Sets the ciphertext CT to be $(\mathbb{A}, C_f)$, where $C_f = [U_0, U_{12}, \ldots, U_{1t_1}, U_1, \ldots, U_{N2}, \ldots, U_{Nt_N}, U_N, V]$.
5. Decrypt: To recover f, user $\mathcal{U}$, whose attributes satisfy CC_i, computes

$$V \oplus H_2\left(\frac{\hat{e}\left(U_0, \frac{n_{\mathbb{A}}}{n_i}\sum_{j=1}^{n_i} SK_{it_i,u,a_{ij}}\right)}{\hat{e}\left(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i} U_i\right)\prod_{j=2}^{t_i} \hat{e}\left(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)}\right)}\right).$$

Observe that:

$$H_2\left(\frac{\hat{e}\left(U_0, \frac{n_{\mathbb{A}}}{n_i}\sum_{j=1}^{n_i} SK_{it_i,u,a_{ij}}\right)}{\hat{e}\left(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i}\right)\prod_{j=2}^{t_i} \hat{e}\left(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)}\right)}\right)$$

$$= H_2\left(\frac{\hat{e}\left(U_0, \frac{n_{\mathbb{A}}}{n_i}\sum_{j=1}^{n_i}\left(SK_{i1} + \sum_{k=2}^{t_i} mk_{i(k-1)} P_{ik} + mk_{it_i} mk_u Pa_{ij}\right)\right)}{\hat{e}\left(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i} U_i\right)\prod_{j=2}^{t_i} \hat{e}\left(U_{ij}, \frac{n_{\mathbb{A}}}{n_i} Q_{i(j-1)}\right)}\right)$$

$$= H_2\left(\frac{\hat{e}(Q_0, n_{\mathbb{A}} rP_1)\prod_{k=2}^{t_i} \hat{e}\left(Q_{i(k-1)}, n_{\mathbb{A}} U_{ik}\right)\hat{e}\left(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i} U_i\right)}{\hat{e}\left(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i} U_i\right)\prod_{j=2}^{t_i} \hat{e}\left(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)}\right)}\right)$$

$$= H_2(\hat{e}(Q_0, n_{\mathbb{A}} rP_1))$$

Note that $V \oplus H_2(\hat{e}(Q_0, n_{\mathbb{A}} rP_1)) = f$, and thus user $\mathcal{U}$ can successfully recover file f.

Remark. To achieve better performance, we enable user $\mathcal{U}$ to send the value of Q-tuple $i_{(t_i-1)}$ to the CSP before decrypting data, so that the CSP can help to calculate the value of $\prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)})$. Given this value, $\mathcal{U}$ executes bilinear map operations two times to recover the file.

6. Performance and security evaluation

6.1. P a c a a

The efficiency of the *Setup*, *CreateDM*, *CreateAttribute* and *CreateUser* algorithms is rather straightforward. Therefore, we only analyze the costs introduced by the *Encryption* algorithm and the *Decryption* algorithm. To encrypt a file f under a DNF access control policy $\mathbb{A} = \bigvee_{i=1}^{N} (CC_i)$, a user needs to compute one bilinear map of Q_0 and P_1, and $O(NT)$ number of exponentiation operations to output a ciphertext of $O(NT)$ length, where $T \in \mathbb{Z}^+$ is the maximum depth of all DMs administering

attributes in $\mathbb{A}$. Notice that the computation for bilinear map of Q_0 and P_1 is independent of the message to be encrypted, and hence can be done once for all. To recover f, a user who possesses user attribute secret keys on all attributes in CC_i needs to execute $O(1)$ bilinear map operations.

In Table 2, we briefly compare the HABE scheme with the work by Bethencourt et al. (2007) and the work by Muller et al. (2008). Since both of them do not support full delegation, we consider the performance of the HABE in the case of the first level DM administering all attributes in a domain. We believe that the most expensive computation is bilinear map operation, abbreviated as *map*, and the next is the exponentiation operation, abbreviated as *exp*.

In Table 2, n is the number of attributes associated with a user, S is the number of attributes in an access structure, N is the number of conjunctive clauses in an access structure, and P is the number of attributes in an access structure that is matched by attributes in a user's private key.

6.2. S d , a a ,

We first provide an intuitive security argument, and prove the HABE scheme to be semantically secure under the random oracle model and the BDH assumption (Boneh and Franklin, 2001) in A. Recall that a confidential file f is encrypted in the form of $f \oplus H_2(\hat{e}(Q_0, rn_{\mathbb{A}}P_1))$. Therefore, an adversary $\mathcal{A}$ needs to construct $\hat{e}(Q_0, rn_{\mathbb{A}}P_1) = \hat{e}(U_0, SK_1)^{n_{\mathbb{A}}}$ to decrypt C_f. According to the constraints in the security game, adversary $\mathcal{A}$ enables none of the users to possess a sufficient set of user attribute secret keys to decrypt C_f.

For ease of presentation, we have the following assumptions: Adversary $\mathcal{A}$ has requested user attribute secret keys for user $\mathcal{U}$ on all but one of the attributes $a_{i1}, \ldots, a_{i(k-1)}, a_{i(k+1)}, \ldots, a_{in_i}$ in CC_i, and for user $\mathcal{U}'$ on the missing attribute a_{ik}. The only occurrence of SK_1 is in the user attribute secret key, so the adversary has to use user attribute secret keys requested for $\mathcal{U}$ and $\mathcal{U}'$ for bilinear map, yielding:

$$\hat{e}\left(U_0, \frac{n_{\mathbb{A}}}{n_i}\sum_{j=1,j\neq k}^{n_i} SK_{it_i,u,a_{ij}} + \frac{n_{\mathbb{A}}}{n_i} SK_{it_i,u',a_{ik}} + \alpha\right)$$
$$= \hat{e}(U_0, SK_1)^{n_{\mathbb{A}}} \prod_{t=2}^{t_i} \hat{e}\left(Qi_{(t-1)}, U_{it}\right)^{n_{\mathbb{A}}} \hat{e}(rP_0, \alpha)\hat{e}\left(mk_{u'}mk_{it_i}P_0, rP_{a_{ik}}\right)^{\frac{n_{\mathbb{A}}}{n_i}}$$
$$\times \hat{e}\left(mk_u mk_{it_i}P_0, r\sum_{j=1,j\neq k}^{n_i} P_{a_{ij}}\right)^{\frac{n_{\mathbb{A}}}{n_i}}$$

Table 2 – Comparisons of CP-ABE schemes.

Properties	Reference (Bethencourt et al., 2007)	Reference (Muller et al., 2008)	Our scheme
User key size	$O(2n)$	$O(n)$	$O(n)$
Ciphertext	$O(2S)$	$O(3N)$	$O(N)$
Encryption (exp)	$O(2N)$	$O(3N)$	$O(N)$
Decryption (map)	$O(2P)$	$O(1)$	$O(1)$
Access structure	Monotone	DNF	DNF

for some α. To obtain $\hat{e}(U_0, SK_1)^{n_{\mathbb{A}}}$, the last four elements have to be eliminated. However, the values of $\hat{e}(mk'_u mk_{it_i}P_0, rP_{a_{ik}})$ and $\hat{e}(mk_u mk_{it_i}P_0, r\sum_{j=1,j\neq k}^{n_i} P_{a_{ij}})$ are unknown to the adversary, and cannot be constructed. Therefore, $\mathcal{A}$ cannot recover the file.

7. Revocation

Revocation of users and keys is a well studied, but a non-trivial problem. In our HABE scheme, when a user $\mathcal{V}$ is revoked, it is imperative to update public keys of attributes in $S_{\mathcal{V}}$, user attribute secret keys for remaining users who possess at least one attribute in $S_{\mathcal{V}}$, and re-encrypt data whose access structure specifies at least one attribute in $S_{\mathcal{V}}$, where set $S_{\mathcal{V}}$ contains all attributes associated with $\mathcal{V}$. If all these tasks are performed by the DMs, it would introduce a heavy computing overhead. Therefore, we propose a scalable revocation scheme, which takes advantage of abundant resources in a cloud by delegating most of the computing tasks in revocation to the CSP.

To enable this revocation scheme to work well, we require each attribute a with ID_a to be bound to a version number, which increases by one whenever a user associated with a is revoked. Correspondingly, each attribute public key is changed into this form: $PK_a^t = (v_a^t, PK_i, ID_a)$, where $t \in \mathbb{Z}_q$ is the version number of the attribute public key, and $v_a^t \in \{0,1\}^*$ is a string corresponding to t. We simply present the scalable revocation scheme by describing the following algorithm:

1. *UpdateAttribute*: DM_i updates PK_a^t to PK_a^{t+1} by adding each version number to 1, and outputs a PRE key by setting $Pkey_a^{t+1} = H_{mk_i}(PK_a^{t+1}) - H_{mk_i}(PK_a^t)$.
2. *CreateUpdateKey*: To generate an update key for updating $PK_a^{t'}$ to the latest public key PK_a^t, the CSP sets $UpdateKey_a = (Pkey_a^{t'+1} + \ldots + Pkey_a^t) = H_{mk_i}(PK_a^t) - H_{mk_i}(PK_a^{t'})$.
3. *ReEncrypt*: Suppose $CT' = (\mathbb{A}', [U'_0, U'_{12}, \ldots, U'_{1t_1}, U'_1, \ldots, U'_{N2}, \ldots, U'_{Nt_N}, U'_N, V'])$ is the original ciphertext, $\{PK_a^{t'}\}$ is a set of overdue public keys of attributes in $\mathbb{A}'$, $\{PK_a^t\}$ is a set of latest attribute public keys corresponding to $\{PK_a^{t'}\}$, and $\{UpdateKey_a\}$ is a set of update keys for updating attribute public keys in $\{PK_a^{t'}\}$ to $\{PK_a^t\}$. The CSP re-encrypts the ciphertext by first setting $V = V'$, $U_0 = U'_0$, and $U_i = U'_i + \sum UpdateKey_a U_0$, where $a \in \{PK_a^{t'}\} \wedge CC'_i$ for $1 \le i \le N$, and then updating overdue public keys of attributes in $\mathbb{A}'$ to the latest version.
4. *UpdateSK*: Suppose $UpdateKey_a$ is the update key for updating $PK_a^{t'}$ to PK_a^t. User $\mathcal{U}$ updates the user attribute secret key by setting $SK_{i,u,a}^t = SK_{i,u,a}^{t'} + UpdateKey_a mk_u mk_i P_0$. Observe that:

$$\begin{aligned} SK_{i,u,a}^t &= SK_{i,u,a}^{t'} + UpdateKey_a mk_u mk_i P_0 \\ &= SK_i + mk_i mk_u P_a^{t'} + \left(H_{mk_i}(PK_a^t) - H_{mk_i}\left(PK_a^{t'}\right)\right) mk_i mk_u P_0 \\ &= SK_i + mk_i mk_u P_a^t \end{aligned}$$

Therefore, the ciphertext re-encrypted by the *ReEncrypt* algorithm under $\{UpdateKey_a | a \in \mathbb{A}\}$ is equal to the ciphertext encrypted by the *Encrypt* algorithm under $\{PK_a^t | a \in \mathbb{A}\}$. The correctness of the revocation scheme is proved.

7.1. S d a a

Data is encrypted with the HABE scheme, which can be proved to be semantically secure under the BDH assumption. Therefore, as long as we can show that the revocation scheme is as secure as the HABE scheme, the security of the revocation scheme is derived. Due to the security of the HABE scheme, the CSP must obtain either master key of proper DM or proper user attribute secret keys to decrypt the ciphertext. As compared to the HABE scheme, the revocation scheme discloses additional PRE keys to the CSP. Note that given these PRE keys, the CSP cannot calculate the DM's master key. Furthermore, without old user attribute secret keys, the CSP cannot calculate the latest user attribute secret keys. Since, we assume that the CSP will not collude with the users, it cannot obtain old user attribute secret keys. Therefore, the security of the revocation scheme is the same as the HABE scheme.

7.2. P a c a a

We only consider the computation cost on the DMs in the user revocation. Suppose $S_{\mathcal{V}}$ denotes the attribute set associated with user $\mathcal{V}$, and $|S_{\mathcal{V}}|$ is the number of attributes in $S_{\mathcal{V}}$. When user $\mathcal{V}$ is revoked, the data owner first runs the *UpdateAttribute* algorithm to update public keys of attributes in $S_{\mathcal{V}}$ to the latest version and generate corresponding PRE keys, which requires $|S_{\mathcal{V}}|$ multiplication operations on $\mathbb{G}_1$. After sending the PRE keys to the CSP, the DM can go off-line. The other algorithms are run by the CSP without the involvement of the DM. Therefore, the revocation scheme, which introduces less computation costs on the DM, provides high scalability.

8. System level description

In this section, we describe the working process of the HABE scheme together with the revocation scheme, at the system level. The process consists of seven components as follows:

1. **System setup**. The TTP runs the *Setup* algorithm to generate system parameters *params* and the root master key MK_0. It then sends *params* along with its signatures on each part of *params* to the CSP and the enterprise user.
2. **Domain setup**. The TTP runs the *CreateDM* algorithm to generate master keys for the ITPs at the next level, which in turn, will generate master keys for the ITPs at the next level. The generated keys along with the creator's signatures are transmitted over a trusted and authenticated channel.
3. **File creation**. We enable each end user to maintain a user attribute list (UAL), which records the latest public key of each attribute that he once used during encryption and decryption. Before sharing a file in cloud servers, the end user processes the file as follows: (1) Define a DNF access control $\mathbb{A}$ for the file. (2) Select a unique ID as the keyword of the file. (3) Encode the file as Bennett et al. (2002), e.g., each file is divided into data blocks of 1 KB size and can be queried using the selected keyword. (4) Encrypt each data block using the *Encrypt* algorithm.

   The CSP maintains a user list (UL), which records all the authorized end users. On receiving an encrypted file, it first verifies if the sender is a valid end user in UL. If true, it duplicates and distributes the ciphertext in cloud servers.
4. **User grant**. When a new end user $\mathcal{U}$ wants to join the system, the leftmost ITP at the second level, denoted $ITP_{\diamond}$, first assigns $\mathcal{U}$ a unique ID, denoted ID_u, and a set of attributes. Then, it generates a private key corresponding to $\mathcal{U}$'s public key using the *Extraction* algorithm in the HIBE system (Gentry and Silverberg, 2002). Finally, it sends all these messages along with its signatures on each component of these messages to $\mathcal{U}$ over a trusted and authenticated channel.

   Furthermore, it also sends ID_u along with its signature on ID_u to the CSP. On receiving ID_u, the CSP adds it to UL in a proper position after verifying the signature.

   On receiving an ID, a set of attributes, and a private key from $ITP_{\diamond}$, $\mathcal{U}$ begins to request secret keys from the ITPs administering any attribute associated with himself.

   On receiving a request for secret keys from $\mathcal{U}$, the ITP runs the *CreateUser* algorithm to generate secret keys for $\mathcal{U}$. Then, it signs the generated keys, encrypts the keys and signatures, under $\mathcal{U}$'s public key using the Encryption algorithm in the HIBE system (Gentry and Silverberg, 2002), and sends the ciphertext to $\mathcal{U}$.

   On receiving the ciphertext, $\mathcal{U}$ first decrypts it with his private key requested from $ITP_{\diamond}$, and then verifies the signatures. If correct, he accepts the secret keys.
5. **User revocation**. The revocation process can be divided into three stages as follows:

   Stage 1. When a user $\mathcal{V}$ is revoked, $ITP_{\diamond}$ first records the ID of the revoked user, denoted ID_v, and then constructs multiple attributes set $S_{i,v,a}$, which contains all attributes administered by ITP_i and possessed by $\mathcal{V}$, for $1 \le i \le n$. Next, it encrypts public keys of attributes in $S_{i,v,a}$ and its signatures on each attribute public key in $S_{i,v,a}$, under ITP_i's public key using the Encryption algorithm in the HIBE system (Gentry and Silverberg, 2002), and sends the ciphertext to ITP_i, for $1 \le i \le n$.

   Each ITP maintains an attribute list (AL), which records the latest public key of each attribute administered by itself. ITP_i, on receiving this message from $ITP_{\diamond}$, first decrypts it, and then verifies the signatures. If true, for the public key of each attribute a in $S_{i,v,a}$, it first checks AL_i to find its position, and then runs the *UpdateAttribute* algorithm to update the attribute public key and outputs a PRE key. Finally, it encrypts the latest public keys and PRE keys of attributes in $S_{i,v,a}$ along with its signatures on each component of these messages, under $ITP_{\diamond}$'s public key using the Encryption algorithm in the HIBE system (Gentry and Silverberg, 2002), and returns the ciphertext to $ITP_{\diamond}$.

   $ITP\diamond$, on collecting all the public keys and PRE keys, first constructs an overall attributes set S_a, which contains all attributes in $S_{i,v,a}$ for $1 \le i \le n$. Then, it signs ID_v, and the latest public keys and corresponding PRE Keys of attributes in S_a, and encrypts these messages along with their signatures, under the CSP's public key using the Encryption algorithm in a HIBE system (Gentry

and Silverberg, 2002). Finally, it sends the ciphertext to the CSP.

The CSP maintains an AHL which records the public key evolution history of each attribute, and the PRE key used. On receiving the ciphertext, it first decrypts it, and then verifies the signatures. If true, it deletes ID_v from UL, and for every attribute in S_a, stores the attribute public key and PRE key, in a proper position in AHL.

Stage 2. By applying PRE and LRE, the CSP can re-encrypt files and update user secret keys in a "lazy" way as follows:

When user $\mathcal{U}$ wants to retrieve a file with access control $\mathbb{A}$, he sends attribute public keys recorded in UAL_u for all attributes in $\mathbb{A}$, together with necessary information to the CSP.

The CSP, on receiving an access request from $\mathcal{U}$, first verifies if $\mathcal{U}$ is a valid end user. If true, it tests whether the version numbers of all attributes in $\mathbb{A}$ are the latest ones. If so, it returns the ciphertext directly; Otherwise, it first re-encrypts the ciphertext using the *ReEncrypt* algorithm. Next, it determines an overdue attributes set $S'_{u,a}$ based on all attribute public keys received from $\mathcal{U}$, and for every attribute in $S'_{u,a}$, runs the *CreateUpdateKey* algorithm to get corresponding $UpdateKey_a$. Finally, it signs the latest public keys and corresponding update keys of all attributes in $S'_{u,a}$, encrypts all these messages along with their signatures, under $\mathcal{U}$'s public key using the *Encryption* algorithm in the HIBE system (Gentry and Silverberg, 2002), and sends this ciphertext and the updated ciphertext of the requested file to $\mathcal{U}$.

Stage 3. On receiving the ciphertext, $\mathcal{U}$ first decrypts it using his private key, and then verifies the signatures. If true, for every attribute a in $S'_{u,a}$, $\mathcal{U}$ runs the *UpdateSK* algorithm to update his attribute secret key.

6. **File access**. This operation consists of the second and third stages of the user revocation operation.
7. **File deletion**. This operation can only be performed at the request of the data owner/sender. To delete a file, the data owner sends $H(H(ID))$ along with his signature on $H(H(ID))$ to the CSP, where ID is a unique keyword of the file. The CSP deletes the data file and all its copies in cloud servers after verifying the signature.

9. Conclusion

In this paper, we proposed a HABE model, a HABE scheme, and a revocation mechanism, so as to simultaneously achieve: (1) high performance; (2) fine-grained access control; (3) scalability; and (4) full delegation, in cloud computing. We describe the scalable revocation scheme from the systematic point of view, and prove the HABE scheme, which is also collusion resistant, to be semantically secure against adaptive chosen plaintext attacks under the BDH assumption and the random oracle model.

In future work, we will design a more expressive encryption scheme, which can be proved to have full security ((under the standard model) (Gentry and Halevi, 2009; Waters, 2009)).

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61073037, 60725208, and 60811130528, Hunan Provincial Science and Technology Program under Grant Nos. 2010GK2003 & 2010GK3005, and Changsha Science and Technology Program under Grant Nos. K 1003064–11 & K 1003062–11.

Appendix A. Security Proof

T 1

Suppose that Algorithm $\mathcal{A}$ is an adversary that has the advantage ϵ of successfully attacking the HABE scheme. Suppose Algorithm $\mathcal{A}$ specifies access policy $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij})$, and makes at most $q_{H_2} > 0$ hash queries to random oracle H_2, and at most $q_E > 0$ user attribute secret key queries in a domain. Then, there is an adversary $\mathcal{B}$ that breaks the BDH problem with the advantage at least $\epsilon' = 2\epsilon N^N / q_{H_2}(e(q_E + N))^N$, and a running time $O(time(\mathcal{A}))$. Here, $e \approx 2.71$ is the base of the natural logarithm.

P

Let H_1, H_2, and H_A be random oracles from $\{0,1\}^*$ to $\mathbb{G}_1$, from $\mathbb{G}_2$ to $\{0,1\}^n$, and from $\{0,1\}^*$ to $\mathbb{Z}_q$ respectively. Algorithm $\mathcal{B}$ is given q, $\mathbb{G}_1$, $\mathbb{G}_2$, $\hat{e}$, P_0, $\mu_0 = \alpha P_0$, $\mu_1 = \beta P_0$, and $\mu_2 = \gamma P_0$, where $< q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} >$ are the outputs of a BDH parameter generator for a sufficiently large security parameter, P_0 is a generator of $\mathbb{G}_1$, and α, β, and γ are random elements of $\mathbb{Z}_q$. The goal is to output $D = e(g,g)^{\alpha\beta\gamma} \in \mathbb{G}_2$. Let D be the solution to the BDH problem. Algorithm $\mathcal{B}$ finds D by interacting with Algorithm $\mathcal{A}$ as follows:

S I

Algorithm $\mathcal{B}$ sets $Q_0 = \mu_0 = \alpha P_0$ and gives Algorithm $\mathcal{A}$ $q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, Q_0, H_1, H_2, H_A >$ as the system parameters, where H_1, H_2, and H_A are controlled by Algorithm $\mathcal{B}$, as described below:

H_1-Queries: Algorithm $\mathcal{B}$ maintains a list of tuples called H_1-List, in which each entry is a tuple of the form $(ID - tuple_j, P - tuple_j, b - tuple_j, mk - tuple_j, c - tuple_j)$. H_1-List is initially empty. When Algorithm $\mathcal{A}$ queries H_1 at a point $ID - tuple_i = (ID_{i1}, \ldots, ID_{it_i})$, Algorithm $\mathcal{B}$ responds as follows: Let y be maximal such that $(ID_{i1}, \ldots, ID_{iy}) = (ID_{j1}, \ldots, ID_{jy})$ for some tuple $((ID_{j1}, \ldots, ID_{jt_j}), (P_{j1}, \ldots, P_{jt_j}), (b_{j1}, \ldots, b_{jt_j}), (mk_{j1}, \ldots, mk_{jt_j}), (c_{j1}, \ldots, c_{jt_j}))$ already in H_1-List. Then:

1. For $1 \le k \le y$, Algorithm $\mathcal{B}$ sets $P_{ik} = P_{jk}$, $b_{ik} = b_{jk}$, $mk_{ik} = mk_{jk}$, and $c_{ik} = c_{jk}$.
2. For $y < k \le t_i$, Algorithm $\mathcal{B}$:
   (a) picks two random elements mk_{ik} and $b_{ik} \in \mathbb{Z}_q$.
   (b) sets $c_{ik} = 1$ and $P_{ik} = b_{ik} P_0$.
   (c) Put $((ID_{i1}, \ldots, ID_{it_i}), (P_{i1}, \ldots, P_{it_i}), (b_{i1}, \ldots, b_{it_i}), (mk_{i1}, \ldots, mk_{it_i}), (c_{i1}, \ldots, c_{it_i}))$ in H_1-List and return $H_1(ID - tuple_i) = (P_{i1}, \ldots, P_{it_i}) \in \mathbb{G}_1$ to Algorithm $\mathcal{A}$

H -Queries: Algorithm $\mathcal{B}$ maintains a list of tuples called H_{mk}-List, in which each entry is a tuple of the form $(ID-tuple_{a_j}, P-tuple_{a_j}, b-tuple_{a_j}, mk-tuple_{a_j}, c-tuple_{a_j})$. H_{mk}-List is initially empty. When Algorithm $\mathcal{A}$ issues attribute secret key queries for a user on attribute a_i with $(ID_{i1}, \ldots, ID_{it_i}, ID_{a_i})$ in Phase 1 or Phase 2, Algorithm $\mathcal{B}$ runs the H_{mk}-Queries as follows:

1. Check if $ID-tuple_{a_i}$ already appears on H_{mk}-List. If so, returns with $H_{mk}(ID-tuple_{a_i}) = (P-tuple_{a_i})$.
2. Otherwise, run the **H_1-Queries** at point of $(ID_{i1}, \ldots, ID_{it_i})$ to obtain appropriate tuple in H_1-List.
3. Pick two random elements mk_{a_i} and $b_{a_i} \in \mathbb{Z}_q$.
4. Pick a random coin $c_{a_i} \in \{0,1\}$ so that $Pr[c_{a_i} = 0] = \delta$ for some δ that will be determined later.
5. If $c_{a_i} = 1$, Algorithm $\mathcal{B}$ sets $P_{a_i} = mk_{it_i} b_{a_i} P_0$.
6. If $c_{a_i} = 0$, Algorithm $\mathcal{B}$ sets $P_{a_i} = b_{a_i} P_0 - mk_{it_i}^{-1} P_1$, where $mk_{it_i}^{-1}$ is the inverse of mk_{it_i} modulo q.
7. Put $((ID_{i1}, \ldots, ID_{it_i}, ID_{a_i}), (P_{i1}, \ldots, P_{it_i}, P_{a_i}), (b_{i1}, \ldots, b_{it_i}, b_{a_i}), (mk_{i1}, \ldots, mk_{it_i}, mk_{a_i}), (c_{i1}, \ldots, c_{it_i}, c_{a_i}))$ in H_{mk}-List and return $H_{mk}(ID-tuple_{a_i}) = (P-tuple_{a_i})$.

H_2-Queries: Algorithm $\mathcal{B}$ maintains a list of tuples called H_2-List, in which each entry is a tuple of the form (T_j, V_j). The list is initially empty.

When Algorithm $\mathcal{A}$ queries H_2 at a point of T_i, Algorithm $\mathcal{B}$ checks if $T_i = T_j$, where T_j already appears on H_2-List in the form of (T_j, V_j). If so, Algorithm $\mathcal{B}$ responds to Algorithm $\mathcal{A}$ with $H_2(T_i) = V_j$. Otherwise, Algorithm $\mathcal{B}$ picks a random string $V_i \in \{0,1\}^n$, adds the tuple (T_i, V_i) to H_2-List, and responds to Algorithm $\mathcal{A}$ with $H_2(T_i) = V_i$.

H_A-Queries: Algorithm $\mathcal{B}$ maintains a list of tuples called H_A-List, in which each entry is a tuple of the form (PK_{u_j}, mk_{u_j}). The list is initially empty. When Algorithm $\mathcal{A}$ issues attribute secret key queries for user $\mathcal{U}_i$ with user public key PK_{u_i} in Phase 1 or Phase 2, Algorithm $\mathcal{B}$ runs the H_A -Queries as follows: Check if $PK_{u_i} = PK_{u_j}$, where PK_{u_j} already appears on H_A-List in the form of (PK_{u_j}, mk_{u_j}). If so, return $H_A(PK_{u_i}) = mk_{u_j}$; Otherwise, pick a random number $mk_{u_i} \in \mathbb{Z}_q$, add the tuple (PK_{u_i}, mk_{u_i}) to H_A -List, and return $H_A(PK_{u_i}) = mk_{u_i}$.

Phase 1: At any time, Algorithm $\mathcal{A}$ may query secret key on any $PK_{a_j} = (ID_{j1}, \ldots, ID_{jt_j}, ID_{a_j})$ for any PK_{u_i}. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the H_A-Queries Algorithm to obtain an appropriate tuple (PK_{u_i}, mk_{u_i}) in H_A-List.
2. Run the H_{mk}-Queries Algorithm to obtain an appropriate tuple $(ID-tuple_{a_j}, P-tuple_{a_j}, b-tuple_{a_j}, mk-tuple_{a_j}, c-tuple_{a_j})$ in H_{mk}-List.
3. If $c_{a_j} = 1$, then Algorithm $\mathcal{B}$ returns P a_j to Algorithm $\mathcal{A}$ and terminates the interaction. Otherwise, we know that $P_{a_j} = b_{a_j} P_0 - mk_{jt_j}^{-1} P_1$.
4. Set $Q-tuple_{j(t_j-1)} = (Q_{j1}, \ldots, Q_{j(t_j-1)})$, where $Q_{jk} = mk_{jk} P_0$ for $1 \le k \le t_j - 1$.
5. Set $SK_{jt_j,u_i} = (Q-tuple_{j(t_j-1)}, mk_{jt_j} mk_{u_i} P_0 + mk_{jt_j} \mu_0)$ and $SK_{jt_j,u_i,a_j} = \sum_{k=2}^{t_j} mk_{j(k-1)} P_{jk} + mk_{jt_j} mk_{u_i} P_{a_j} + mk_{jt_j} b_{a_j} \mu_0$, and return P a_j and $(SK_{jt_j,u_i}, SK_{jt_j,u_i,a_j})$ to Algorithm $\mathcal{A}$.

Although Algorithm $\mathcal{B}$ does not know the values of α, it can output a correct attribute secret key for PK_{u_i} at $ID-tuple_{a_j}$ as follows:

By definition, the value of SK_{jt_j,u_i,a_j} should be: $SK'_{j1} + \sum_{k=2}^{t_j} mk'_{j(k-1)} P_{jk} + mk'_{jt_j} mk'_{u_i} P_{a_j}$, where symbol ' denotes authentical keys. Observe that:

$$\begin{aligned} SK_{jt_j,u_i,a_j} &= SK'_{j1} + \sum_{k=2}^{t_j} mk'_{j(k-1)} P_{jk} + mk'_{jt_j}\ mk'_{u_i} P_{a_j} \\ &= \alpha P_{j1} + \sum_{k=2}^{t_j} mk_{j(k-1)} P_{jk} + mk_{jt_j} mk_{u_i} P_{a_j} \\ &\quad + mk_{jt_j} \alpha \left(b_{a_j} P_0 - mk_{jt_j}^{-1} P_{j1} \right) \\ &= \sum_{k=2}^{t_j} mk_{j(k-1)} P_{jk} + mk_{jt_j} mk_{u_i} P_{a_j} + mk_{jt_j} b_{a_j} \mu_0 \end{aligned}$$

Therefore, $(SK_{jt_j,u_i}, SK_{jt_j,u_i,a_j})$ is the correct keys.

Challenge: Once Algorithm $\mathcal{A}$ decides that Phase 1 is over, it outputs an access policy $\mathbb{A}$ that can be expressed as $\mathbb{A} = \bigvee_{i=1}^{N} (CC_i) = \bigvee_{i=1}^{N} (\bigwedge_{j=1}^{n_i} a_{ij})$ and two plaintexts f_0, f_1 on which it wishes to be challenged. Algorithm $\mathcal{B}$ responds as follows:

1. For $1 \le i \le N$: Run the H_{mk}-Queries to obtain the appropriate tuples $(ID-tuple_{a_{ij}}, P-tuple_{a_{ij}}, b-tuple_{a_{ij}}, mk-tuple_{a_{ij}}, c-tuple_{a_{ij}})$ in H_{mk}-List.
2. If $c_{a_{ij}} = 0$ in either of N conjunctive clauses, then Algorithm $\mathcal{B}$ reports failure to Algorithm $\mathcal{A}$ and terminates the interaction. Otherwise, we know that $P_{ik} = b_{ik} P_0$ for $2 \le k \le t_i$, and there are at least one attribute a_{iA*} with $P_{a_{iA*}} = mk_{it_i} b_{a_i A*} P_0$ where $1 \le i \le N$. Therefore, there is at least one attribute secret key in either of conjunctive clauses which has not been queried by Algorithm $\mathcal{A}$ in Phase 1.
3. Pick a random $b \in \{0,1\}$ and a random string $J \in \{0,1\}^n$, and give the ciphertext $CT = (\mathbb{A}, [U_0, U_{12}, \ldots, U_{1t_1}, U_1, \ldots, U_{N2}, \ldots, U_{Nt_N}, U_N, V]) = (\mathbb{A}, [\mu_2, b_{12}\mu_2, \ldots, b_{1t_1}\mu_2, b_{a_{1A*}}\mu_2, \ldots, b_{N_2\mu_2}, \ldots, b_{Nt_N\mu_2}, b_{a_{NA*}\mu_2}, J])$ to Algorithm $\mathcal{A}$.

Note that this challenge implicitly defines: $J = f_b \oplus H_2(\hat{e}(\gamma\mu_0, P_1))$. In other words:

$$J = f_b \oplus H_2(\hat{e}\ (\gamma\alpha P_0, b_1 P_0 + \beta P_0)) = f_b \oplus H_2\left(\hat{e}\,(P_0, P_0)^{\alpha\gamma(\beta + b_1)}\right).$$

By definition, user $\mathcal{U}$, who possesses all attribute secret keys in CC_i, can decrypt the ciphertext using $SK'_{it_i,u}$ and $SK'_{it_i,u,a_{i*}}$, where symbol ' denotes authentic keys. He computes

$$J \oplus H_2\left(\frac{\hat{e}\left(U_0, SK'_{it_i,u,a_{iA*}}\right)}{\hat{e}\left(mk'_{it_i} mk'_u P_0, U_i\right) \prod_{k=2}^{t_i} \hat{e}\left(U_{ik}, Q'_{i(k-1)}\right)} \right)$$

to recover the file f_b. Observe that:

$$J \oplus \mathrm{H}_2\left(\frac{\hat{e}\left(\mathrm{U}_0, \mathrm{SK}'_{\mathrm{it_i},u,a_{i_*}}\right)}{\hat{e}\left(mk'_{\mathrm{it_i}} mk'_u P_0, \mathrm{U_i}\right) \prod_{k=2}^{t_i} \hat{e}\left(\mathrm{U_{ik}}, \mathrm{Q}'_{i(k-1)}\right)}\right)$$

$$= J \oplus \mathrm{H}_2\left(\frac{\hat{e}\left(\mathrm{U}_0, \mathrm{SK}'_{i1} + \sum_{k=2}^{t_i} mk_{i(k-1)} P_{ik} + mk_{\mathrm{it_i}}(mk_u + \alpha) P_{a_{i_*}}\right)}{\hat{e}\left(mk_{\mathrm{it_i}}(mk_u + \alpha) P_0, b_{a_{i_*}} \mu_2\right) \prod_{k=2}^{t_i} \hat{e}\left(\mathrm{U_{ik}}, \mathrm{Q}'_{i(k-1)}\right)}\right)$$

$$= J \oplus \mathrm{H}_2\left(\frac{\hat{e}(\gamma P_0, \alpha P_1) \prod_{k=2}^{t_i} \hat{e}\left(\mathrm{U_{ik}}, \mathrm{Q}'_{i(k-1)}\right) e\left(\gamma P_0, mk_{\mathrm{it_i}}(mk_u + \alpha) P_{a_{i_*}}\right)}{\hat{e}\left(mk_{\mathrm{it_i}}(mk_u + \alpha) P_0, b_{a_{i_*}} \gamma P_0\right) \prod_{k=2}^{t_i} \hat{e}\left(\mathrm{U_{i_k}}, \mathrm{Q}'_{i(k-1)}\right)}\right)$$

$$= J \oplus \mathrm{H}_2\left(\frac{\hat{e}(\gamma P_0, \alpha P_1) e\left(\gamma P_0, mk_{\mathrm{it_i}}(mk_u + \alpha) P_{a_{i_*}}\right)}{\hat{e}\left(\gamma P_0, mk_{\mathrm{it_i}}(mk_u + \alpha) P_{a_{i_*}}\right)}\right)$$

$$= J \oplus \mathrm{H}_2(\hat{e}(\gamma P_0, \alpha \mathrm{P_{i1}}))$$

$$= J \oplus \mathrm{H}_2(\hat{e}(\gamma \mu_0, P_1))$$

Hence, CT is a valid ciphertext for f_b, as required.

Phase 2. Algorithm $\mathcal{A}$ can continue issuing more attribute secret key queries other than $\mathrm{ID} - \mathrm{tuple}_{a_{1_*}}$, …, $\mathrm{ID} - \mathrm{tuple}_{a_{N_*}}$. Algorithm $\mathcal{B}$ responds as in Phase 1.

Guess: Algorithm $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b. At this point, Algorithm $\mathcal{B}$ picks a random pair (T_i, V_i) from H_2-List, and outputs $T_i / \hat{e}(\mu_0, \mu_2)^{b_1}$ as the solution to D.

To complete the proof of Theorem 1, we now show that Algorithm $\mathcal{B}$ correctly outputs D with the probability at least $2\epsilon N^N / q_{H_2}(e(q_E + N))^N$. In the first place, we calculate the probability that Algorithm $\mathcal{B}$ does not abort during the simulation. Suppose Algorithm $\mathcal{A}$ makes a total of q_E attribute secret key queries. Then, the probability that Algorithm $\mathcal{B}$ does not abort in Phase 1 or 2 is δ^{q_E}. In addition, the probability that Algorithm $\mathcal{B}$ does not abort during the challenge step is $(1-\delta)^N$. Therefore, the probability that Algorithm $\mathcal{B}$ does not abort during the simulation is $\delta^{q_E} \cdot (1-\delta)^N$. This value is maximized at $\delta_{opt} = 1 - N/(q_E + N)$. Using δ_{opt}, the probability that Algorithm $\mathcal{B}$ does not abort is at least $(N/e(q_E + N))^N$.

In the second place, we calculate the probability that Algorithm $\mathcal{B}$ outputs the correct result in case that Algorithm $\mathcal{B}$ does not abort. Let Q be the event that Algorithm $\mathcal{A}$ issues a query for V. If ¬Q, we know that the decryption of the ciphertext is independent of Algorithm $\mathcal{A}$'s view. Let $Pr[b = b']$ be the probability that Algorithm $\mathcal{B}$ outputs the correct result, therefore, in the real attack, $Pr[b = b' | \neg Q] = 1/2$. Since Algorithm $\mathcal{A}$ has the advantage ϵ, $|Pr[b = b' | \neg Q] - 1/2| \geq \epsilon$. By deduction, we know $Pr[Q] \geq 2\epsilon$ in the real attack.

Now we know that Algorithm $\mathcal{A}$ will issue a query for V with the probability at least 2ϵ. That is to say, the probability that V appears in some pair on H_2-List is at least 2ϵ. Algorithm $\mathcal{B}$ will choose the correct pair with the probability at least $1/q_{H_2}$, thus Algorithm $\mathcal{B}$ produces the correct answer with the probability at least $2\epsilon/q_{H_2}$. Since Algorithm $\mathcal{B}$ does not abort with the probability at least $(N/e(q_E + N))^N$, we see that Algorithm $\mathcal{B}$'s success probability is at least $\epsilon' = 2\epsilon N^N / q_{H_2}(e(q_E + N))^N$, as required.

REFERENCES

Bennett AK, Grothoff C, Horozov T, Patrascu I. Efficient sharing of encrypted data. In: Proceedings of ACISP. LNCS, vol. 2384; 2002. p. 107–20.

Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Proceedings of ISSP; 2007. p. 321–34.

Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography. In: Proceedings of EUROCRYPT; 1998. p. 127–44.

Boneh D, Boyen X. Efficient Selective-ID secure identity based encryption without random oracles. In: Proceedings of EUROCRYPT. LNCS, vol. 3027; 2004. p. 223–38.

Boneh D, Franklin M. Identity-based encryption from the Weil pairing. In: Proceedings of CRYPTO. LNCS, vol. 2139; 2001. p. 213–29.

Boneh D, Boyen X, Goh E. Hierarchical identity based encryption with constant size ciphertext. In: Proceedings of EUROCRYPT. LNCS, vol. 3494; 2005. p. 440–56.

Gentry C, Halevi S. Hierarchical identity based encryption with Polynomially many levels. In: Proceedings of TCC. LNCS, vol. 5444; 2009. p. 437–56.

Gentry C, Silverberg A. Hierarchical ID-based cryptography. In: Proceedings of ASIACRYPT. LNCS, vol. 2501; 2002. p. 548–66.

Goyal V, Pandey O, Sahai A, Waters B. Attribute-Based encryption for fine-grained access control of encrypted data. In: Proceedings of CCS; 2006. p. 89–98.

Haclgiimfi H, Iyer B, Li C, Mehrotra S. Executing SQL over encrypted data in database-service-provider model. In: Proceedings of SIGMOD; 2002. p. 216–27.

Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K. Plutus: scalable secure file sharing on untrusted storage. In: Proceedings of FAST; 2003. p. 29–42.

Muller S, Katzenbeisser S, Eckert C. Distributed attribute-based encryption. In: Proceedings of ICISC; 2008. p. 20–36.

Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. In: Proceedings of CCS; 2007. p. 195–203.

Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of EUROCRYPT. LNCS, vol. 3494; 2005. p. 457–73.

Wang G, Liu Q, Wu J. Achieving fine-grained access control for secure data Sharing on cloud servers, Wiley's concurrency and computation: Practice and experience, Published online in Wiley Online Library (wileyonlinelibrary.com). doi:10.1002/cpe.1698.

Wang G, Liu Q, Wu J. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of CCS-2010 (Poster), pp. 735–737.

Waters B. Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Proceedings of CRYPTO. LNCS, vol. 5677; 2009. p. 619–36.

Yu S, Wang C, Ren K. Attribute based data sharing with attribute revocation. In: Proceeding of ASIACCS; 2010a. p. 261–70.

Yu S, Wang C, Ren K, Lou W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of INFOCOM; 2010b. p. 15–9.

Guojun Wang received B.Sc. in Geophysics, M.Sc. in Computer Science, and Ph.D. in Computer Science, from Central South University, China. He is Chair and Professor of Department of Computer Science at Central South University. He is Director of Trusted Computing Institute at Central South University. He has been an Adjunct Professor at Temple University, USA; a Visiting Scholar at Florida Atlantic University, USA; a Visiting Researcher at the University of Aizu, Japan; and a Research Fellow at the Hong Kong Polytechnic University. His research interests include network and information security, Internet of things, and cloud computing. He is a senior member of CCF, and a member of IEEE, ACM, and IEICE.

Qin Liu received B.Sc. in Computer Science in 2004 from Hunan Normal University, China, and M.Sc. in Computer Science in 2007 from Central South University, China. She is a Ph.D. candidate at School of Information Science and Engineering, Central South University, China. She is currently a Visiting Student at Temple University, USA. Her research interests include security and privacy issues in cloud computing.

Jie Wu is Chair and Professor at the Department of Computer and Information Sciences at Temple University, USA. He is an IEEE Fellow. He is on the editorial board of IEEE Transactions on Mobile Computing. He was a Distinguished Professor in Department of Computer Science and Engineering, Florida Atlantic University. He served as a Program Director at US NSF from 2006 to 2008. He has been on the editorial board of IEEE Transactions on Parallel and Distributed Systems. He has served as a Distinguished Visitor of the IEEE Computer Society, and is the chairman of the IEEE Technical Committee on Distributed Processing (TCDP). His research interests include wireless networks and mobile computing, parallel and distributed systems, and fault-tolerant systems.

Minyi Guo received Ph.D. in Computer Science from the University of Tsukuba, Japan. Before 2000, he had been a Research Scientist of NEC Corp., Japan. He is now Chair and Distinguished Professor of Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. He was a Professor at the School of Computer Science and Engineering, University of Aizu, Japan. His research interests include parallel and distributed processing, parallelizing compilers, pervasive computing, embedded software optimization, molecular computing, and software engineering. He is a senior member of IEEE, a member of the ACM, IPSJ and IEICE.