CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2000; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Achieving Fine-grained Access Control for Secure Data Sharing on Cloud Servers

Guojun Wang[1,*], Qin Liu[1] and Jie Wu[2]

[1] *School of Information Science and Engineering, Central South University, Changsha, Hunan Province, P. R. China, 410083*
[2] *Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA*

SUMMARY

With more and more enterprises sharing their sensitive data on cloud servers, building a secure cloud environment for data sharing has attracted a lot of attention in both industry and academic communities. In this paper, we propose a conjunctive precise and fuzzy identity-based encryption (PFIBE) scheme for secure data sharing on cloud servers, which allows the encryption of data by specifying a recipient identity (ID) set, or a disjunctive normal form (DNF) access control policy over attributes, so that only the user whose ID belonging to the ID set or attributes satisfying the DNF access control policy can decrypt the corresponding data. Our design goal is to propose a novel encryption scheme, which simultaneously achieves a fine-grained access control, flexibility, high performance, and full key delegation, so as to help enterprise users to enjoy more secure, comprehensive, and flexible services. We achieve this goal by first combining the hierarchical identity-based encryption (HIBE) system and the ciphertext-policy attribute-based encryption (CP-ABE) system, and then marking each user with both an ID and a set of descriptive attributes, finally separating the access control policy into two parts: a recipient ID set and a DNF attribute-based access control policy.

KEY WORDS: *Secure Storage; Cloud Computing; Precise and Fuzzy Identity-Based Encryption*

*Correspondence to: School of Information Science and Engineering, Central South University, Changsha, Hunan Province, P. R. China, 410083; E-mail: csgjwang@mail.csu.edu.cn

Contract/grant sponsor: National Natural Science Foundation of China; contract/grant number: 61073037, 90718034 & 60773013; Hunan Provincial Science and Technology Program; contract/grant number: 2010GK2003 & 2010GK3005; Changsha Science and Technology Program; contract/grant number: K1003064-11 & K1003062-11

1. INTRODUCTION

Cloud computing [1], as one of current most exciting technology areas, denotes an architectural shift towards thin clients and scalably centralized provision of computing and storage resources on-demand. By combining emerging techniques, such as virtualization and service-oriented computing, three types of servers are available in a pay-as-you-go manner, i.e., infrastructure as a service (IaaS), where users make use of a cloud service provider's (CSP's) computing, storage, and networking infrastructure to deploy any arbitrary software, e.g., Amazon EC2; platform as a service (PaaS), where users deploy user-created or acquired applications written with programming languages and tools supported by CSPs, e.g., Microsoft Windows Azure; and software as a service (SaaS), where users make use of CSPs' software running on a cloud infrastructure, e.g, Google Docs.

Obviously, moving data into a cloud offers great convenience to users since they can access data in a cloud anytime and anywhere, using any device, without caring about the large capital investment in both the deployment and management of hardware infrastructures. Especially for small and medium enterprises with limited budgets, they can achieve cost savings and productivity enhancements by using cloud-based services to manage projects, collaborate on documents and presentations, manage enterprisewide contacts and schedules, and the like. However, allowing a CSP, operated for making a profit, to take care of confidential enterprise data, raises underlying security and privacy issues that will result in a huge loss for enterprises. For instance, an untrustworthy CSP may sell the confidential information about an enterprise to the enterprise's closest business competitors for profit. Therefore, a natural way to keep sensitive data confidential against an untrusted CSP is to encrypt outsourced data in advance.

We consider the following application scenario: Company A, whose hierarchy is shown in Fig. 1, pays a CSP for sharing data on cloud servers, where any member in Company A can store files on cloud servers, and retrieve authorized files anytime and anywhere using any device. Suppose Bob wants to store an encrypted file on cloud servers, so that only the members satisfying the access control policy, shown in Fig. 2, can access it.

The intuition behind this access control policy is that the file should only be accessed by Bob, who is the boss of the enterprise, Alice in the sales department (SD), whose ID is "2010", Clark in the financial department (FD), whose ID is "2011", Donald in the personnel department (PD), whose ID is "2012", and the department manager, system analysts, or senior programmers in the research and development department (RDD). Furthermore, the party that administers all members' IDs, is superior to the party that administers attributes "isBoss", "inRDD", "DepartmentManager", "SystemAnalyst", and "SeniorProgrammer".

In the above application scenario, (1) the encrypted file involves more than one recipient; (2) the encrypter wishes to specify a fine-grained access control policy, which describes the intended recipients using not only their IDs but also descriptive attributes; (3) the recipients wish to access the file anytime and anywhere using any device, such as thin clients with limited bandwidth, CPU, and memory capabilities; (4) corresponding to the hierarchy of Company A, a delegation mechanism in the generation of keys inside the enterprise is needed.

Therefore, it is needed to propose a secure data sharing scheme, which simultaneously achieves (1) "one-to-many encryption", (2) flexibility, (3) high performance, and (4) full key delegation, so as to be more applicable in cloud computing. Our design goal is to propose a

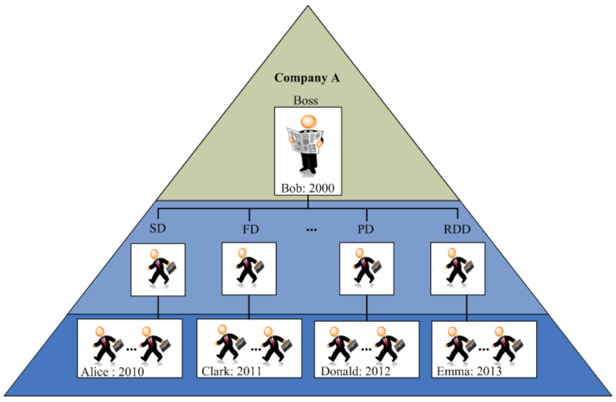
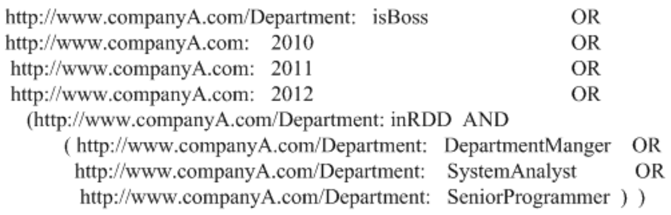


Figure 1. The hierarchy of Company A



```
http://www.companyA.com/Department:   isBoss                     OR
http://www.companyA.com:   2010                                  OR
 http://www.companyA.com:   2011                                 OR
 http://www.companyA.com:   2012                                 OR
   (http://www.companyA.com/Department: inRDD  AND
        ( http://www.companyA.com/Department:   DepartmentManger    OR
          http://www.companyA.com/Department:   SystemAnalyst        OR
           http://www.companyA.com/Department:   SeniorProgrammer  )  )
```

Figure 2. Sample access control policy. This access control policy can be expressed as a Boolean formula over IDs and attributes. Either an attribute or an ID consists of a web site specifying which party administers it and an identifier describing 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.

novel encryption scheme, so as to help enterprise users to enjoy more secure, comprehensive, and flexible services. Our contributions are fourfold:

1. A conjunctive precise and fuzzy identity-based encryption (PFIBE) model, which creatively combines a HIBE system and a CP-ABE system, is better applicable to enterprises outsourcing their data for sharing environment.
2. A PFIBE scheme, based on the proposed model, which achieves a fine-grained access control over a set of IDs and descriptive attributes, enables users to enjoy more comprehensive and high-quality services.
3. The PFIBE scheme, which requires only a constant number of bilinear map operations during decryption, is tailored for best serving the needs of accessing data anytime and anywhere.

4. The PFIBE scheme, which supports full key delegation within an enterprise, can better embody the hierarchy of enterprises.

Organization. The rest of this paper is organized as follows: We begin with a discussion of related work in Section 2, and an introduction to preliminaries in Section 3. Then, we provide an overview and an efficient construction for the PFIBE scheme in Section 4 and 5, respectively. Next, we analyze the performance and security for the PFIBE scheme in Section 6. Finally, we conclude this paper in Section 7.

2. RELATED WORK

In this section, we introduce some preliminary cryptographic primitives to be used to meet our requirements.

2.1. Traditional symmetric/public key cryptosystem

In a traditional symmetric key cryptosystem (SKC), a shared key between a sender and all recipients is used as an encryption key and a decryption key. When a sender wants to encrypt a file to n recipients, he first chooses a unique shared key corresponding to the file and sends it to all recipients in a secure way, and then encrypts the file using the shared key. Finally, the corresponding ciphertext is stored in a cloud. Therefore, the key size will grow linearly with the number of ciphertexts.

In a traditional public key cryptosystem (PKC), each user has a public/private key pair, and messages encrypted with a recipient's public key can only be decrypted with the corresponding private key. When a sender wants to encrypt a file to n recipients, he first obtains the authenticated public keys of all the recipients, and then encrypts the file using each recipient's public key, respectively. Finally, the n copies of the corresponding ciphertexts are stored in a cloud. Therefore, both the computational cost for encryption, and the length of the ciphertexts, are proportional to the total number of intended recipients.

Obviously, neither of them should be applied directly while sharing data on cloud servers, since they are inefficient to encrypt a file to multiple recipients, and fail to support attribute-based access control and key delegation.

2.2. Broadcast Encryption

Fiat et al [26] first introduced the concept of the broadcast encryption (BE), in which a broadcaster encrypts a message for some subset S of users who are listening on a broadcast channel, so that only the recipients in S can use their private keys to decrypt the message. The first proposal of a BE system is secure against a collusion of k users, which means that such a scheme may be insecure if more than k users collude.

Recently, the study of a BE system, dedicated to having full collusion resistance and better performance, has become more and more important with the ever-increasing concern with copyright issues and the increasing interest in secure multicasting over cable television and the Internet. Compared with the attribute-based encryption (ABE) system, the performance of the BE system may be worse. As discussed in Goyal et al [15], the efficiency of the systems proposed in Boneh et al [28] and Halevy et al [27], which are the best known broadcast encryption schemes, is not only dependent on the size of the authorized user set, but also requires the broadcaster to refer to its database of user authorizations.

In a BE system, there are only two parties: a broadcaster and multiple users, where the broadcaster generates the secret keys for all the users, and can broadcast an encrypted message to some subset of the users. Obviously, a BE system achieves "one-to-many encryption" with general performance, however, it may not applied directly while sharing data on cloud servers, since it fails to support attribute-based access control and key delegation.

2.3. Hierarchical Identity-Based Encryption

Shamir [4] proposed the idea of identity-based cryptography, but a fully functional identity-based encryption (IBE) scheme was not found until recent work by Boneh et al [6] and Cocks [5]. An IBE scheme is a PKC, where any arbitrary string corresponding to a unique user information is a valid public key. The corresponding private key is computed by a trusted third party (TTP) called the private key generator (PKG). Compared with the traditional PKC, the IBE system eliminates online look-ups for the recipient's authenticated public key, but introduces the key escrow problem.

In an IBE system, there is only one PKG to distribute private keys to each user, which is undesirable for a large network because the PKG has a burdensome job. Horwitz et al [7], dedicated to reducing the workload on the root PKGs, introduced the concept of a HIBE system. They constructed a concrete two-level HIBE scheme, in which a root PKG needed only to generate private keys for domain-level PKGs that, in turn generated private keys for all the users in their domains at the next level. Their scheme, with total collusion resistance on the upper level and partial collusion resistance on the lower level, has chosen ciphertext security in the random oracle model.

Gentry et al [11] proposed a HIBE scheme (from here-on referred to as the G-HIBE scheme) with total collusion resistance at an arbitrary number of levels, which has chosen ciphertext security in the random oracle model under the BDH assumption. It is worth noticing that the G-HIBE possesses a "valuable" property, i.e., an encrypted file can be decrypted by a recipient and all his ancestors, using their own secret keys, respectively, which can be seen as "one-to-many encryption" in a sense. A subsequent construction by Boneh et al [9] 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 in encryption and decryption, grows linearly with the depth of a recipient in the hierarchy. For better performance, Boneh et al [10] proposed an efficient HIBE system, which requires only a constant length of ciphertext and a constant number of bilinear map operations in decryption with selective-ID security under the BDH assumption without random oracles.

In recent work, Gentry et al [11] proposed a fully secure HIBE scheme by using identity-based broadcast encryption with key randomization; Waters [12] achieved full security in systems under simple assumption by using dual system encryption. Among others, by making use of the "valuable" property of the G-HIBE scheme, Liu et al [13] proposed an efficient sharing of the secure cloud storage services (ESC) scheme, where a sender can specify several users as the recipients for an encrypted file by taking the number and public keys of the recipients as inputs of a HIBE system. The limitation of their scheme is that the length of ciphertexts grows linearly with the number of recipients, so that it can only be used in the case that a confidential file involves a small set of recipients.

The HIBE system naturally achieves key delegation, and some HIBE schemes achieve "one-to-many encryption" with general performance, however, it may not be applied directly while sharing data on cloud servers, since it fails to support an attribute-based access control.

2.4. Attribute-Based Encryption

An ABE scheme is actually a generalization of the IBE scheme: In an IBE system, a user is identified by only one attribute, i.e., the ID. Sahai et al [14] first introduced the concept of the ABE schemes, in which a sender encrypted a message, specifying an attribute set and a number d, so that only a recipient who has at least d attributes of the given attributes can decrypt the message. Their scheme, which is referred to as threshold encryption, is collusion resistant and has selective-ID security.

Based on their work, Goyal et al [15] proposed a fine-grained access control ABE scheme, which supports any monotonic access formula consisting of AND, OR, or threshold gates. 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 ciphertexts. A subsequent construction by Ostrovsky et al [16] allows for non-monotonic access structures (also include NOT gates, i.e., negative constraints in a key's access formula).

Bethencourt et al [17] introduced a ciphertext-policy ABE (CP-ABE) scheme, in which the roles of the ciphertexts and keys are reversed in contrast with the KP-ABE scheme: The access structure is specified in the ciphertext, while the private key is simply created with respect to an attributes set. Muller et al [18] introduced a distributed attribute-based encryption (DABE), and provided an efficient construction that requires a constant number of bilinear map operations in decryption. The limitation of the DABE scheme is that the access control policy must be expressed as a DNF, and the number of bilinear map operations in encryption and the length of ciphertexts is directly proportional to the number of conjunctive clauses in the DNF. Both of the above mentioned CP-ABE schemes provide a proof of security in the generic bilinear group model and the random oracle model.

In recent work, Chase [19] provided a construction for a multi-authority ABE system, where each authority would administer a different domain of attributes. Chase et al [20] provided a more practice-oriented multi-authority ABE system, which removes the trusted central authority while preserving user privacy. Among others, Yu et al [21] exploited and uniquely combined techniques of ABE, proxy re-encryption (PRE) [24], and lazy re-encryption (LRE) [25] to delegate most of the computation tasks involved in user revocation to untrusted CSPs without disclosing the underlying data contents, which may make a KP-ABE system more applicable in a cloud environment. Since each file is associated with an access control rather than a set of attributes as KP-ABE, it is harder to delegate the re-encryption operation to a third party. We find that the proxy re-encryption technique is only applied to CP-ABE until in recent work of Wang et al [22] and Yu et al [23].

The ABE system, which was first designed to achieve fault tolerance, can achieve attribute-based access control with better performance compared with the above cryptographic primitives. However, it may not be adopted directly while sharing data on cloud servers, since it fails to support ID-based access control and "full key delegation". It is worth noticing that, although, some ABE systems support delegation between users, which enables a user to generate attribute secret keys containing a subset of his own attribute secret keys for other users, we hope to achieve a full delegation mechanism, that is, a delegation mechanism between attribute authorities (AAs), which independently make decisions on the structure and semantics of their attributes. In the HIBE system, a user private key, which is extracted from keys of his parent, can also be used to extract private keys for the users at the next level. In the ABE system, the attribute authority generates only a set of attribute secret keys, without the ability to extract a low-level master key as HIBE systems using its own master key for a user, and thus fail to support full delegation.

Summary. The characteristics of a HIBE system and an ABE system are supporting "full delegation" and "fine-grained access control over attributes", respectively. Furthermore, the HIBE system is designed for encrypting to an exact recipient, as well as, the ABE system is designed for encrypting to a set of attributes. Therefore, to simultaneously achieve fine-grained access control over IDs and attributes, and full key delegation within an enterprise, a natural way is to combine such encryption models. This is a non-trivial task, until we find that the "valuable" property of the G-HIBE scheme, since this "one-to-many" property can be regarded as a meeting point with an ABE system. Therefore, we first construct public/secret keys as Gentry et al [8], which are the intuitions of the "one-to-many" property. Then, we mark each user with an ID and a set of descriptive attributes. Finally, inspired by Muller et al [18], we separate the access control policy into two parts: a recipient ID set and a DNF attribute-based access control policy. We find that an encryption scheme achieves not only better performance, but also the combination of a HIBE system and a CP-ABE system, by expressing the access control policy in such a form.

3. PRELIMINARIES

In this section, we first introduce some related definitions and complexity assumptions, then outline the G-HIBE scheme, and finally describe our system model and security model.

3.1. Definitions and Assumptions

We introduce some related definitions and complexity assumptions, which closely follows those in Boneh et al [6].

Definition 3.1 (Bilinear Map): Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ is a multiplicative group. A bilinear map, $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, satisfies the following properties:

1. Computable: There is a polynomial time algorithm to compute $\hat{e}(P, Q) \in \mathbb{G}_2$, for any $P, Q \in \mathbb{G}_1$.

2. Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}_q^*$.
3. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$.

Definition 3.2 (BDH Parameter Generator): A randomized algorithm $\mathcal{IG}$ is called a BDH parameter generator if $\mathcal{IG}$ takes a sufficiently large security parameter $K > 0$ as input, runs in polynomial time in K, and outputs a prime number q, the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and the description of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

Definition 3.3 (BDH Problem): Given a random element $P \in \mathbb{G}_1$, as well as aP, bP, and cP, for some $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc} \in \mathbb{G}_2$.

Definition 3.4 (BDH Assumption): If $\mathcal{IG}$ is a BDH parameter generator, the advantage $Adv_{\mathcal{IG}}(\mathcal{B})$ that an algorithm $\mathcal{B}$ has in solving the BDH problem is defined to be the probability that $\mathcal{B}$ outputs $\hat{e}(P, P)^{abc}$ on inputs q, $\mathbb{G}_1$, $\mathbb{G}_2$, $\hat{e}$, P, aP, bP, cP, where $< q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} >$ are the outputs of $\mathcal{IG}$ for a sufficiently large security parameter K, P is a random element $\in \mathbb{G}_1$, and a, b, c are random elements of $\mathbb{Z}_q^*$. The BDH assumption is that $Adv_{\mathcal{IG}}(\mathcal{B})$ is negligible for any efficient algorithm $\mathcal{B}$.

3.2. OUTLINE OF THE G-HIBE SCHEME

To reveal the insights behind the "valuable" property that enable a sender to encrypt a file to multiple recipients (the recipient and all his ancestors), we will simply outline the G-HIBE scheme.

In the G-HIBE scheme, the user public key is an ID-tuple, which consists of his ID and his ancestors' IDs. Let User$_t$ be a user whose public key is ID-tuple$_t$ = (ID$_1, \ldots,$ ID$_t$). User$_t$'s ancestors are the root PKG and the lower-level PKGs whose ID-tuples are (ID$_1, \ldots,$ ID$_i$) for $1 \leq i < t$. The G-HIBE scheme consists of five randomized polynomial time algorithms as follows:

1. *Root Setup:* The Root PKG first picks $mk_0 \in \mathbb{Z}_q$ as the root master key, and then chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, two random oracles H_1: $\{0, 1\}^* \to \mathbb{G}_1^*$ and H_2: $\mathbb{G}_2 \to \{0, 1\}^n$ for some n, and a random generator $P_0 \in \mathbb{G}_1$. Let $Q_0 = mk_0 P_0$. The system parameters are $params =< q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, Q_0, H_1, H_2 >$.
2. *Lower-level Setup:* User$_t$ picks a random element $mk_t \in \mathbb{Z}_q$ as his master key.
3. *Extraction:* User$_t$'s parent generates User$_t$'s private key SK_t by computing $S_t = S_{t-1} + mk_{t-1} P_t$ and giving $(Q_1, \ldots, Q_{t-1})$, where S_{t-1} and mk_{t-1} are the secret point and master key of User$_t$'s parent, respectively, $Q_i = mk_i P_0$ for $1 \leq i \leq t-1$, and $P_t = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$.
4. *Encryption:* To encrypt a file f to User$_t$, the sender first picks a random element $r \in \mathbb{Z}_q^*$, and then sets $C_f = [U_0, U_2, \ldots, U_t, V] = [rP_0, rP_2, \ldots, rP_t, f \oplus H_2(\hat{e}(Q_0, rP_1))]$, where $P_i = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ for $1 \leq i \leq t$.
5. *Decryption:* Given the ciphertext C_f, User$_t$ computes $V \oplus H_2(\hat{e}(U_0, S_t) / \prod_{i=2}^{t} \hat{e}(Q_{i-1}, U_i))$ to recover f.

In summary, a user public key is an ID-tuple, which consists of the user's ID and the user's ancestors' public keys; a user private key consists of a secret point and a Q-tuple, both of which are generated from keys of his parent; a sender takes system parameters and the recipient's public key as the inputs of *Encryption* algorithm, so that the intended recipient can decrypt the file using his own private key.

Liu et al [13] find the "valuable" property of the G-HIBE scheme as follows: Let User_i with $(\text{ID}_1, \ldots, \text{ID}_i)$, where $1 \leq i < t$ be one of User_t's ancestors. Note that: $S_t = S_i + \sum_{j=i+1}^{t} mk_{j-1}P_j$.

For $1 \leq i < t$:

$$\begin{aligned}
&\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^{t} \hat{e}(Q_{i-1}, U_i)} \\
&= \frac{\hat{e}(U_0, S_i + \sum_{j=i+1}^{t} mk_{j-1}P_j)}{\prod_{k=2}^{i} \hat{e}(Q_{k-1}, U_k) \prod_{k=i+1}^{t} \hat{e}(Q_{k-1}, U_k)} \\
&= \frac{\hat{e}(U_0, S_i) \prod_{j=i+1}^{t} \hat{e}(Q_{j-1}, U_j)}{\prod_{k=2}^{i} \hat{e}(Q_{k-1}, U_k) \prod_{k=i+1}^{t} \hat{e}(Q_{k-1}, U_k)} \\
&= \frac{\hat{e}(U_0, S_i)}{\prod_{k=2}^{i} \hat{e}(Q_{k-1}, U_k)}
\end{aligned}$$

Therefore, given a part of the ciphertext $[\mathbb{A}, U_0, U_2, \ldots, U_i, V]$, User_i can recover f by computing $V \oplus H_2(\hat{e}(U_0, S_i) / \prod_{k=2}^{i} \hat{e}(Q_{k-1}, U_k))$.

The insights behind such a "valuable" property is that a user can recover f using his own private key, in case that his public key is on inputs of the *Encryption* algorithm, due to the construction of user public key, user private key, and the encryption algorithm. By using such a "valuable" property, Liu et al [13] proposed an ESC scheme, where a sender takes the number of recipients and the public keys of recipients as inputs during encryption, so that the intended recipients can decrypt the file using their own private key.

In this paper, also by making use of the "valuable" property of the G-HIBE scheme, we first mark each domain master (DM) and attribute with a unique ID, but mark each user with both an ID and a set of descriptive attributes. Specifically, to resist collusion, we enable user secret keys, including user private key, user identity secret key, and user attribute secret key, to relate to be his unique ID. Then, as the G-HIBE scheme, we enable an entity's secret key to be extracted from the DM administering itself, and an entity's public key, which denotes its position in the model, to be an ID-tuple consisting of the public key of the DM administering itself and its ID. Next, we separate the access control policy into two parts: a recipient ID set and a DNF attribute-based access control policy. Specifically, to achieve a DNF attribute-based access control policy in encryption, we substitute "+" operation for "AND" semantics, and separating character "," for "OR" semantics, so that only users who possess all attribute secret keys in at least one of the conjunctive clauses can decrypt the file.

3.3. System Model

We assume that the system is composed of the following parties: a CSP, a trusted third party (TTP), enterprise users, end users, and internal trusted parties (ITPs). The former two parties are easily understood: A CSP operates an amount of interconnected cloud servers with abundant storage capacity and computation power to provide high-quality services, and a 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 enterprise data

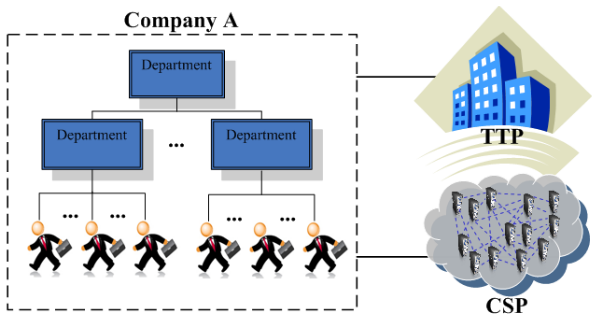


Figure 3. System Model

on cloud servers is an enterprise user, all personnel in the company who share data on cloud servers are end users, a department in the company that delegates keys inside the company is an ITP.

3.4. Security Model

As described in Haclgiimfi et al [3], there are two main attacks under such a circumstance, i.e., outer attacks initiated by unauthorized outsiders, and inner attacks initiated by an honest, but curious CSP, as well as some curious end users. Here, we call a CSP more interested in file contents than other secret information an honest, but curious CSP, and an end user, who is ineligible to decrypt a file, but wants to read it as a curious end user. A CSP might collude with curious end users for the purpose of harvesting file contents. We assume that the communication channels are secured under existing security protocols, such as SSL.

4. OVERVIEW OF THE PROPOSED SCHEME

In this section, we first propose a PFIBE model, and then provide a summary of keys used in our scheme. Next, we outline the PFIBE scheme based on the proposed model. Finally, we provide the security definition of the proposed scheme.

4.1. The PFIBE Model

Corresponding to the system model, the PFIBE 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 is the TTP, and the domains are enterprise users. More precisely, a domain consists of many domain masters (DMs) corresponding to ITPs, and numerous users corresponding to end users. Over authenticated and trusted channels, the RM distributes secret

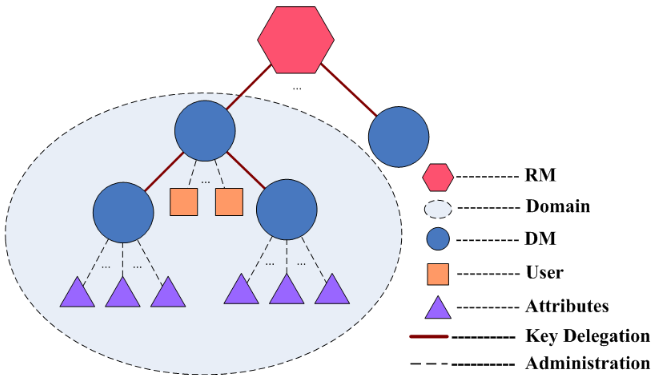


Figure 4. PFIBE Model

keys to DMs, which, in turn distribute secret keys to DMs at the next level and users in their domains. In a domain, the DM hierarchy depends on the scale of an enterprise, e.g., a two-level PFIBE model is enough in the case of a small enterprise.

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 the AA in a CP-ABE system, is responsible for delegating a generation of keys to DMs at the next level and distributing secret keys to users. Specially, we enable first-level DM, denoted $\mathrm{DM}_{\diamond}$, 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 full control over the structure and semantics of their attributes.

4.2. Summary of Keys

In our scheme, there are multiple keys with different usages. Therefore, we provide the summary of the most relevant keys used in our scheme (see Table I) to provide a quick reference:

1. The root master key MK_0, possessed by the RM, is used to generate secret keys for the top-level DMs.
2. DM_i, with ID_i, possesses a public key PK_i and a master key MK_i. PK_i is an ID-tuple of the form $(\mathrm{PK}_{i-1}, \mathrm{ID}_i)$ where PK_{i-1} is the public key of DM_i's parent, except for $\mathrm{PK}_{\diamond} = (\mathrm{ID}_{\diamond})$. PK_i can be used to generate personalized MK_i which can be used in the generation of user keys. It is worth noticing that only $\mathrm{DM}_{\diamond}$ is responsible for generating user private keys, and other DMs are responsible for generating user identity secret keys and user attribute secret keys.

Table I. Summary of keys

Key	Description	Usage
MK_0	Root master key	Creation of master key for DMs at the first level
$\mathrm{PK}_\Diamond$	$\mathrm{DM}_\Diamond$'s public key	Creation of $\mathrm{MK}_\Diamond$
$\mathrm{MK}_\Diamond$	$\mathrm{DM}_\Diamond$'s master key	Creation of user private key
$\mathrm{PK}_i(i \neq \Diamond)$	DM_i's public key	Creation of MK_i
$\mathrm{MK}_i(i \neq \Diamond)$	DM_i's master key	Creation of user private key, 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_u	$\mathcal{U}$'s private key requested from $\mathrm{DM}_\Diamond$	Decryption
$\mathrm{SK}_{i,u}$	$\mathcal{U}$'s identity secret key requested from DM_i	Decryption
$\mathrm{SK}_{i,u,a}$	$\mathcal{U}$'s attribute secret key requested from DM_i on attribute a	Decryption
PK_a	a's public key	Creation of user attribute secret key

3. User $\mathcal{U}$, with ID_u and a set of descriptive attributes $\{a\}$, possesses a user public key PK_u, a user private key SK_u, a set of user identity secret keys $\{\mathrm{SK}_{i,u}\}$, and a set of user attribute secret keys $\{\mathrm{SK}_{i,u,a}\}$. PK_u is an ID-tuple of the form $(\mathrm{PK}_\Diamond, \mathrm{ID}_u)$ where $(\mathrm{PK}_\Diamond)$ is the public key of the first-level DM. PK_u is used by the $\mathrm{DM}_\Diamond$ to generate personalized SK_u, and by the DM_i to generate personalized $\mathrm{SK}_{i,u}$ and $\{\mathrm{SK}_{i,u,a}\}$, all of which can be used in the decryption.
4. Attribute a, with ID_a, possesses a public key PK_a that is used in the encryption and in the generation of attribute secret keys. PK_a is an ID-tuple of the form $(\mathrm{PK}_i, \mathrm{ID}_a)$ where PK_i is the public key of $\mathrm{DM}_i(i \neq \Diamond)$ administering attribute a.

4.3. The PFIBE Scheme

Based on the PFIBE model, we provide a formal definition of the PFIBE scheme. For ease of presentation, we illustrate how the PFIBE scheme works by the following application scenario (see Fig. 5), which will be used as our sample application for the rest of this paper.

Let all the departments and personnel in Company A constitute a domain, denoted Dom_A, where $\mathrm{DM}_\Diamond$ is the first level DM. Suppose, in Dom_A, there are M users $\mathrm{U}_1, \ldots, \mathrm{U}_M$, whose public keys are in the form of $\mathrm{PK}_{u_i} = (\mathrm{PK}_\Diamond, \mathrm{ID}_{u_i})$ for $1 \leq i \leq M$, and m DMs $\mathrm{DM}_\Diamond, \ldots, \mathrm{DM}_m$, whose public keys are in the form of $\mathrm{PK}_i = (\mathrm{PK}_{i-1}, \mathrm{ID}_i)$ for $2 \leq i \leq m$, except for $\mathrm{PK}_\Diamond = (\mathrm{ID}_\Diamond)$.

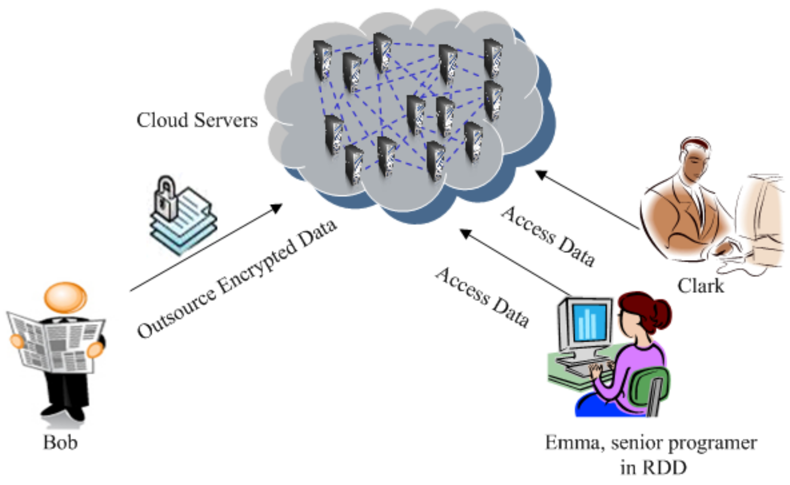


Figure 5. Sample Application Scenario

http://www.companyA.com/Department: isBoss	OR
(http://www.companyA.com/Department: DepartmentManager	AND
http://www.companyA.com/Department: inRDD)	OR
(http://www.companyA.com/Department SystemAnalyst	AND
http://www.companyA.com/ Department: inRDD)	OR
(http://www.companyA.com/Department: SeniorProgrammer	AND
http://www.companyA.com/Department: inRDD)	

Figure 6. DNF attribute-based access control policy. The DNF policy consists of four conjunction clauses, in each of which all attributes are administered by the same DM. A user needs all attribute secret keys in at least one of the conjunctive clauses to be able to read it.

When Bob wants to encrypt a confidential file to some one who satisfies an access control policy, he first separates it into two parts: a recipient ID set $\mathbb{R}$, and a DNF attribute-based access control $\mathbb{A}$. To illustrate how this transformation works, we return to the previously mentioned application scenario. The access control policy in Fig. 1 can be transformed into two parts: $\mathbb{R}$={"http://www.companyA.com:2010","http://www.companyA.com:2011","http://www.companyA.com:2012"}, and a DNF attribute-based access control $\mathbb{A}$ as shown in Fig. 6.

Suppose $\mathbb{R} = \{\text{ID}_{u_1}, \ldots, \text{ID}_{u_n}\}$ where $1 \le n \le M$, and $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij})$, 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 $(\text{ID}_{i1}, \ldots,$

ID_{it_i}) be the DM administering all attributes in CC_i, where ID_{ik} for $1 \leq k < t_i$ are IDs of DM_{it_i}'s ancestors and ID_{i1}=$\mathrm{ID}_{\diamondsuit}$ is the ID of $\mathrm{DM}_{\diamondsuit}$.

Next, Bob sends the following message to a CSP:

$$\mathrm{MSG}_{Bob2CSP} = \mathit{Encrypt}(params, f, \mathbb{R}, \mathbb{A}, \{\mathrm{PK}_u|\mathrm{ID}_u \in \mathbb{R}\}, \{\mathrm{PK}_a|\mathrm{ID}_a \in \mathbb{A}\})$$

where *params* are the system parameters, f is a confidential file, $\mathbb{R}$ is the recipient ID set, $\mathbb{A}$ is a DNF attribute-based access control policy, $\{\mathrm{PK}_u\}$ are the public keys of partial intended recipients, and $\{\mathrm{PK}_a\}$ are the public keys of the attributes describing the remaining recipients.

User $\mathcal{U}$, whose ID belonging to $\mathbb{R}$ or attributes satisfying the *i-th* conjunctive clause CC_i in $\mathbb{A}$, can recover f by computing the following formulae (1) and (2), respectively:

$$\mathit{RDecrypt}(params, \mathrm{CT}, \mathrm{SK}_u) \tag{1}$$

$$\mathit{ADecrypt}(params, \mathrm{CT}, \mathrm{SK}_{it_i,u}, \{\mathrm{SK}_{it_i,u,a}\}), \tag{2}$$

where CT is the ciphertext, and SK_u, $\mathrm{SK}_{it_i,u}$, and $\{\mathrm{SK}_{it_i,u,a}\}$ are $\mathcal{U}$'s private key, identity secret key, and attribute secret keys of all attributes in CC_i, respectively.

Definition 4.1 (Definition of the PFIBE Scheme): The PFIBE scheme consists of seven randomized polynomial time algorithms as follows:

1. $\mathit{Setup}(K) \rightarrow (params, \mathrm{MK}_0)$: The RM takes a sufficiently large security parameter K as input, and outputs system parameters *params* and root master key MK_0.
2. $\mathit{CreateDM}(params, \mathrm{MK}_i, \mathrm{PK}_{i+1}) \rightarrow (\mathrm{MK}_{i+1})$: Whether the RM or a DM generates master keys for the DMs directly under it using *params* and its master key.
3. $\mathit{CreateSK}(params, \mathrm{MK}_{\diamondsuit}, \mathrm{PK}_u) \rightarrow (\mathrm{SK}_u)$: $\mathrm{DM}_{\diamondsuit}$ first checks whether $\mathcal{U}$ is eligible for PK_u. If so, it generates a user private key using *params* and its master key; Otherwise, it outputs "NULL".
4. $\mathit{CreateUser}(params, \mathrm{MK}_i, \mathrm{PK}_u, \mathrm{PK}_a) \rightarrow (\mathrm{SK}_{i,u}, \mathrm{SK}_{i,u,a})$: A DM except for the $\mathrm{DM}_{\diamondsuit}$ 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".
5. $\mathit{Encrypt}(params, f, \mathbb{R}, \mathbb{A}, \{\mathrm{PK}_u|\mathrm{ID}_u \in \mathbb{R}\}, \{\mathrm{PK}_a|\mathrm{ID}_a \in \mathbb{A}\}) \rightarrow (\mathrm{CT})$: A user takes a file f, a recipient ID set $\mathbb{R}$, a DNF attribute-based access control policy $\mathbb{A}$, public keys of recipients whose IDs belonging to $\mathbb{R}$, and public keys of all attributes whose IDs belonging to $\mathbb{A}$, as inputs, and outputs a ciphertext CT.
6. $\mathit{RDecrypt}(params, \mathrm{CT}, \mathrm{SK}_u) \rightarrow (f)$: A user, whose ID belonging to $\mathbb{R}$, takes *params*, the ciphertext, and the user private key, as inputs, to recover the plaintext.
7. $\mathit{ADecrypt}(params, \mathrm{CT}, \mathrm{SK}_{i,u}, \{\mathrm{SK}_{i,u,a}|a \in CC_j\}) \rightarrow (f)$: A user, whose attributes satisfy the *j-th* conjunctive clause CC_j in $\mathbb{A}$, takes *params*, the ciphertext, the user identity secret key, and the user attribute secret keys of all attributes in CC_j, as inputs, to recover the plaintext.

4.4. Semantic Security of the PFIBE scheme

We define security for the PFIBE scheme in the sense of semantic security [6]. Semantic security captures our insight that given a ciphertext, the adversary learns nothing about the

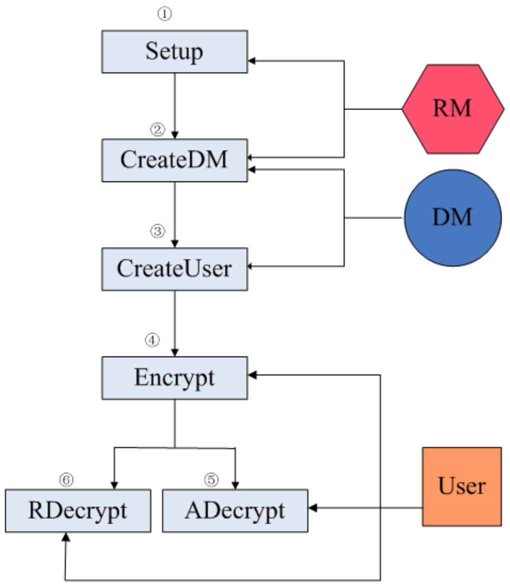


Figure 7. Working Process of the PFIBE scheme

corresponding plaintext, thus we also say that a semantically secure scheme is IND-CPA secure [6]. As those in Boneh et al [6] did, we strengthen the standard definition of semantic security for the PFIBE scheme by allowing an adversary to query any user key, including user private key and user attribute secret keys in $\mathrm{Dom}_{\mathcal{A}}$, of his choice. Also, we allow an adversary to choose the recipient ID set $\mathbb{R}$ and the DNF attribute-based access control policy $\mathbb{A}$ on which it wishes to be challenged. Even under such an attack, the adversary should not be able to distinguish an encryption of a file f_0 from an encryption of a file f_1 so long as he did not obtain any of the private keys corresponding to $\mathbb{R}$ and all the attribute secret keys in any conjunctive clause in $\mathbb{A}$. We define the semantic security for the PFIBE scheme in terms of a game as follows:

Setup: The challenger runs the *Setup* algorithm when inputting a sufficiently large security parameter K to generate the system parameters *params* and a root master key. It gives adversary $\mathcal{A}$ *params*, but keeps the root master key to itself.

Phase 1: Adversary $\mathcal{A}$ can query any user key in $\mathrm{Dom}_{\mathcal{A}}$ of his choice. When adversary $\mathcal{A}$ issues a query for user $\mathcal{U}$'s private key, the challenger first runs the *CreateDM* algorithm to generate keys for $\mathrm{DM}_{\diamond}$, and then runs the *CreateSK* algorithm to generate a private key for $\mathcal{U}$. When adversary $\mathcal{A}$ issues a query for user $\mathcal{U}$ at attribute a, which is administered by DM_i, the challenger first runs the *CreateDM* algorithm to generate keys for DM_i, and then runs the *CreateUser* algorithm to generate an identity secret key $\mathrm{SK}_{i,u}$ and an attribute secret key $\mathrm{SK}_{i,u,a}$ for $\mathcal{U}$. These queries may be asked adaptively.

Challenge: Once adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs a recipient ID set $\mathbb{R} = \{\mathrm{ID}_{u_1}, \ldots, \mathrm{ID}_{u_n}\}$, a DNF attribute-based access control policy $\mathbb{A} = \overset{N}{\underset{i=1}{\vee}} (CC_i) = \overset{N}{\underset{i=1}{\vee}} (\overset{n_i}{\underset{j=1}{\wedge}} a_{ij})$, and two equal length plaintexts f_0, f_1 on which it wishes to be challenged. The constraints are: (1) None of the ID $\in \mathbb{R}$ appears in any private key query in Phase 1; (2) None of the users, for whom adversary $\mathcal{A}$ requests secret keys in Phase 1, possess all attribute secret keys in one of the conjunctive clauses in $\mathbb{A}$. The challenger picks a random bit $b \in \{0,1\}$, sets $C_{f_b} = \mathit{Encrypt}(\mathit{params}, f_b, \mathbb{R}, \mathbb{A}, \{\mathrm{PK}_u | \mathrm{ID}_u \in \mathbb{R}\}, \{\mathrm{PK}_a | \mathrm{ID}_a \in \mathbb{A}\})$, and sends C_{f_b} as the challenge to $\mathcal{A}$.

Phase 2: Adversary $\mathcal{A}$ issues more user attribute secret key queries with the same constraints as those in Challenge. The challenger responds as that in Phase 1.

Guess: Adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, and wins the game if $b = b'$. We define $\mathcal{A}$'s advantage in breaking the PFIBE scheme to be $Adv_{\mathcal{A}}(K) = |Pr[b = b'] - 1/2|$.

Definition 4.2 (Semantic Security of the PFIBE Scheme): We say that the PFIBE scheme is semantically secure if for any polynomial time adversary $\mathcal{A}$, the function $Adv_{\mathcal{A}}(K)$ is negligible.

5. CONSTRUCTION OF THE PFIBE SCHEME

In this section, we construct the PFIBE scheme using the bilinear map. Let $\mathcal{IG}$ be a BDH parameter generator. We present the PFIBE scheme by describing the following seven randomized polynomial time algorithms:

1. $\mathit{Setup}(K) \rightarrow (\mathit{params}, \mathrm{MK}_0)$: The RM first picks $mk_0 \in \mathbb{Z}_q$, and then chooses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, two random oracles H_1: $\{0,1\}^* \rightarrow \mathbb{G}_1$ and H_2: $\mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n, and a random generator $P_0 \in \mathbb{G}_1$. Let $Q_0 = mk_0 P_0 \in \mathbb{G}_1$. The system parameters $\mathit{params} = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, Q_0, H_1, H_2)$ will be publicly available, while $\mathrm{MK}_0 = (mk_0)$ will be kept secret.
2. $\mathit{CreateDM}(\mathit{params}, \mathrm{MK}_i, \mathrm{PK}_{i+1}) \rightarrow (\mathrm{MK}_{i+1})$: To generate the master key for DM_{i+1} with PK_{i+1}, the RM or DM_i first picks a random element $mk_{i+1} \in \mathbb{Z}_q$, and then computes $\mathrm{SK}_{i+1} = \mathrm{SK}_i + mk_i P_{i+1}$ where $P_{i+1} = H_1(\mathrm{PK}_{i+1}) \in \mathbb{G}_1$, and $Q_{i+1} = mk_{i+1} P_0 \in \mathbb{G}_1$, finally sets $\mathrm{MK}_{i+1} = (mk_{i+1}, \mathrm{SK}_{i+1}, \text{Q-tuple}_{i+1})$ where $\text{Q-tuple}_{i+1} = (\text{Q-tuple}_i, Q_{i+1})$, and gives the random oracle $H_A : \{0,1\} \rightarrow \mathbb{Z}_q$ that is chosen by the RM and shared in Dom_A. Here, we assume that SK_0 is an identity element of $\mathbb{G}_1$, and $\text{Q-tuple}_0 = (Q_0)$.
3. $\mathit{CreateSK}(\mathit{params}, \mathrm{MK}_\diamondsuit, \mathrm{PK}_u,) \rightarrow (\mathrm{SK}_u)$: To generate a private key for user $\mathcal{U}$ with PK_u, $\mathrm{DM}_\diamondsuit$ first checks whether $\mathcal{U}$ is eligible for PK_u. If so, it first sets $\mathrm{SK}_u = (\text{Q-tuple}_\diamondsuit, \mathrm{SK}_\diamondsuit + mk_\diamondsuit P_u)$, where $P_u = H_1(\mathrm{PK}_u) \in \mathbb{G}_1$. Otherwise, it outputs "NULL".
4. *CreateUser*: To generate a secret key for user $\mathcal{U}$ with PK_u on attribute a with PK_a, DM_i first checks whether $\mathcal{U}$ is eligible for a, and a is administered by itself. If so, it first computes $mk_u = H_A(\mathrm{PK}_u) \in \mathbb{Z}_q$, and then sets $\mathrm{SK}_{i,u} = (\text{Q-tuple}_{i-1}, mk_i mk_u P_0)$, and $\mathrm{SK}_{i,u,a} = \mathrm{SK}_i + mk_i mk_u P_a \in \mathbb{G}_1$, where $P_a = H_1(\mathrm{PK}_a) \in \mathbb{G}_1$; Otherwise, it outputs "NULL".
5. $\mathit{Encrypt}(\mathit{params}, \mathbb{R}, \mathbb{A}, \{\mathrm{PK}_u | \mathrm{ID}_u \in \mathbb{R}\}, \{\mathrm{PK}_a | \mathrm{ID}_a \in \mathbb{A}\}, f) \rightarrow (\mathrm{CT})$: Given a recipient ID set $\mathbb{R} = \{\mathrm{ID}_{u_1}, \ldots, \mathrm{ID}_{u_n}\}$, and a DNF attribute-based access control policy $\mathbb{A} =$

$\overset{N}{\underset{i=1}{\vee}}(CC_i) = \overset{N}{\underset{i=1}{\vee}}(\overset{n_i}{\underset{j=1}{\wedge}} a_{ij})$, where $N \in \mathbb{Z}^+$ is the number of conjunctive clause 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 $(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i})$ be the DM at level t_i, 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}=$\mathrm{ID}_{\diamondsuit}$ is the ID of $\mathrm{DM}_{\diamondsuit}$. The sender:

(a) Computes $P_{\diamondsuit} = H_1(\mathrm{PK}_{\diamondsuit}) \in \mathbb{G}_1$.
(b) Computes $P_{u_i} = H_1(\mathrm{PK}_{u_i}) \in \mathbb{G}_1$, for $1 \le i \le n$.
(c) For $1 \le i \le N$: Computes $P_{ij} = H_1(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{ij}) \in \mathbb{G}_1$ for $1 \le j \le t_i$, and $P_{a_{ij}} = H_1(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i}, \mathrm{ID}_{a_{ij}}) \in \mathbb{G}_1$ for $1 \le j \le n_i$.
(d) 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 $U_0 = rP_0$, $U_{u_1} = rP_{u_i}$, $\ldots$, $U_{u_n} = rP_{u_n}$, $U_{12} = rP_{12}$, $\ldots$, $U_{1t_1} = rP_{1t_1}$, $U_1 = r\sum_{j=1}^{n_1} P_{a_{1j}}$, $\ldots$, $U_{N2} = rP_{N2}$, $\ldots$, $U_{Nt_N} = rP_{Nt_N}$, $U_N = r\sum_{j=1}^{n_N} P_{a_{Nj}}$, and $V = f \oplus H_2(\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit}))$.
(e) Sets the ciphertext to be CT= $(\mathbb{R}, \mathbb{A}, C_f)$, where $C_f = [U_0, U_{u_1}, \ldots, U_{u_n}, U_{12}, \ldots, U_{1t_1}, U_1, \ldots, U_{N2}, \ldots, U_{Nt_N}, U_N, V]$.

6. *RDecrypt*$(params, \mathrm{CT}, \mathrm{SK}_u) \to (f)$: User $\mathcal{U}_i$ whose ID belonging to $\mathbb{R}$, sets $n_{\mathbb{A}}$ to be the LCM of $n_1, \ldots, n_N$, and computes $V \oplus H_2(\hat{e}(n_{\mathbb{A}}U_0, \mathrm{SK}_{\diamondsuit} + mk_{\diamondsuit}P_{u_i})/\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, U_{u_i}))$ to recover f. Observe that:

$$\begin{aligned}
&V \oplus H_2(\hat{e}(n_{\mathbb{A}}U_0, \mathrm{SK}_{\diamondsuit} + mk_{\diamondsuit}P_{u_i})/\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, U_{u_i}))\\
=&V \oplus H_2(\hat{e}(rn_{\mathbb{A}}P_0, mk_0P_{\diamondsuit} + mk_{\diamondsuit}P_{u_i})/\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, rP_{u_i}))\\
=&V \oplus H_2(\hat{e}(rn_{\mathbb{A}}P_0, mk_0P_{\diamondsuit})\hat{e}(rn_{\mathbb{A}}P_0, mk_{\diamondsuit}P_{u_i})/\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, rP_{u_i}))\\
=&V \oplus H_2(\hat{e}(mk_0P_0, rn_{\mathbb{A}}P_{\diamondsuit})\hat{e}(n_{\mathbb{A}}mk_{\diamondsuit}P_0, rP_{u_i})/\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, rP_{u_i}))\\
=&V \oplus H_2(\hat{e}(mk_0P_0, rn_{\mathbb{A}}P_{\diamondsuit})\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, rP_{u_i})/\hat{e}(n_{\mathbb{A}}Q_{\diamondsuit}, rP_{u_i}))\\
=&V \oplus H_2(\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit})) = f
\end{aligned}$$

as required.

7. *ADecrypt*$(params, \mathrm{CT}, \mathrm{SK}_{it_i,u}, \{\mathrm{SK}_{it_i,u,a_{ij}} | 1 \le j \le n_i\}) \to (f)$: User $\mathcal{U}$, whose attributes satisfy CC_i, computes $V \oplus H_2(\frac{\hat{e}(U_0, \frac{n_{\mathbb{A}}}{n_i}\sum_{j=1}^{n_i} \mathrm{SK}_{it_i,u,a_{ij}})}{\hat{e}(mk_u mk_{it_i}P_0, \frac{n_{\mathbb{A}}}{n_i}U_i)\prod_{j=2}^{t_i}\hat{e}(U_{ij}, n_{\mathbb{A}}Q_{i(j-1)})})$ to recover

f. Observe that:

$$
\begin{aligned}
& V \oplus H_2\left(\frac{\hat{e}(U_0, \frac{n_{\mathbb{A}}}{n_i} \sum_{j=1}^{n_i} \mathrm{SK}_{it_i,u,a_{ij}})}{\hat{e}(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)})}\right) \\
&= V \oplus H_2\left(\frac{\hat{e}(U_0, \frac{n_{\mathbb{A}}}{n_i} \sum_{j=1}^{n_i} (\mathrm{SK}_{i1} + \sum_{k=2}^{t_i} mk_{i(k-1)} P_{ik} + mk_{it_i} mk_u P_{a_{ij}}))}{\hat{e}(mk_u mk_{it_i} P_0, \frac{n_{\mathbb{A}}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, \frac{n_{\mathbb{A}}}{n_i} Q_{i(j-1)})}\right) \\
&= V \oplus H_2\left(\frac{\hat{e}(U_0, n_{\mathbb{A}} \mathrm{SK}_{i1}) \hat{e}(U_0, n_{\mathbb{A}} \sum_{k=2}^{t_i} mk_{i(k-1)} P_{ik} + \frac{n_{\mathbb{A}}}{n_i} mk_{it_i} mk_u \sum_{j=1}^{n_i} P_{a_{ij}})}{\hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_{\mathbb{A}}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)})}\right) \\
&= V \oplus H_2\left(\frac{\hat{e}(Q_0, n_{\mathbb{A}} r P_{i1}) \prod_{k=2}^{t_i} \hat{e}(Q_{i(k-1)}, n_{\mathbb{A}} U_{ik}) \hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_{\mathbb{A}}}{n_i} U_i)}{\hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_{\mathbb{A}}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_{\mathbb{A}} Q_{i(j-1)})}\right) \\
&= V \oplus H_2(\hat{e}(Q_0, n_{\mathbb{A}} r P_{i1})) \\
&= V \oplus H_2(\hat{e}(Q_0, n_{\mathbb{A}} r P_{\diamondsuit}))
\end{aligned}
$$

as required.

Remark 1. 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, $\mathcal{U}$ runs the *ADecrypt* algorithm which executes the bilinear map operations for two times to recover the file.

Remark 2. $\mathcal{U}$'s public key $(\mathrm{ID}_{\diamondsuit}, \mathrm{ID}_u)$ and a's public key $(\mathrm{ID}_{\diamondsuit}, \ldots, \mathrm{ID}_i, \mathrm{ID}_a)$ can be concatenated into strings "$\mathrm{ID}_{\diamondsuit} : \mathrm{ID}_u$" and "$\mathrm{ID}_{\diamondsuit} : \ldots : \mathrm{ID}_i : \mathrm{ID}_a$", respectively (" : " denotes string concatenation), both of which are standard inputs of H_1 and H_A hash functions.

Remark 3. All DMs in Dom$_A$ share the H_A hash function, and thus they can generate the same value of mk_u for user $\mathcal{U}$. However, mk_u, which contributes nothing to decryption, will not be given to $\mathcal{U}$.

This concludes the description of the PFIBE scheme.

6. PERFORMANCE AND SECURITY

In this section, we will analyze the performance and security of the proposed PFIBE scheme.

6.1. Performance Analysis

The efficiency of the *Setup*, *CreateDM*, *CreateSK*, *CreateAttribute*, and *CreateUser* algorithms is rather straightforward. The *Setup* algorithm requires only one exponentiation operation to generate system parameters of constant length; The *CreateDM* algorithm requires only

Table II. Comparisons of CP-ABE Schemes

Properties	CP-ABE in ISSP 07	DABE in ICISC 08	Our Scheme
User Key Size	$O(2L)$	$O(L)$	$O(L+I)$
Ciphertext	$O(2S)$	$O(3N)$	$O(NT+n)$
Encryption (exp)	$O(2N)$	$O(3N)$	$O(NT+n)$
Decryption (map)	$O(2P)$	$O(1)$	$O(1)$
Full Delegation	NO	NO	YES
Multiple AAs	NO	YES	YES
Access control over IDs or attributes	NO	NO	YES

two exponentiation operations to generate keys of $O(i)$ length to DMs at Level $i \in \mathbb{Z}^+$; The *CreateSK* algorithm requires $\text{DM}_\diamond$ to execute only one exponentiation operation to generate secret keys of $O(1)$ length to a user; The *CreateUser* algorithm requires DMs at level i to execute $O(L)$ number of exponentiation operations to generate secret keys of $O(i+L)$ length to a user associating with $L \in \mathbb{Z}^+$ attributes.

To encrypt a file f under a recipient ID set $\mathbb{R} = \{\text{ID}_{u_1}, \ldots, \text{ID}_{u_n}\}$ and a DNF access control policy $\mathbb{A} = \overset{N}{\underset{i=1}{\vee}}(CC_i)$, a user needs to compute one bilinear map of Q_0 and P_1, and $O(n+NT)$ number of exponentiation operations to output a ciphertext of $O(n+NT)$ length, where $n \in \mathbb{Z}^+$ is the number of recipients in $\mathbb{R}$, $N \in \mathbb{Z}^+$ is the number of the conjunctive clauses in $\mathbb{A}$, and $T \in \mathbb{Z}^+$ is the maximum depth of all DMs administering attributes in $\mathbb{A}$. Notice that the computation for the 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 whose ID belonging to $\mathbb{R}$ runs the *RDecrypt* algorithm, which needs to execute $O(1)$ bilinear map operations, and a user who possesses user attribute secret keys on all attributes in CC_i runs the *ADecrypt* algorithm, which needs to execute $O(1)$ bilinear map operations.

In Table II, we briefly compare our scheme with the work by Bethencourt et al [17] that is a CP-ABE scheme of a monotone access control and the work by Muller et al [18] that is the CP-ABE scheme of best performance. We believe that the most expensive computation is the bilinear map operation, abbreviated as *map*. The next is the exponentiation operation, abbreviated as *exp*.

In Table II, L, I, S, N, T, n, P denote, the number of attributes associated with a user, the maximum depth of DMs administering attributes associated with a user, the number of attributes in access control, the number of conjunctive clauses in access control, the maximum depth of DMs administering attributes in access control, the number of recipients, and the number of attributes matched by a user's private key attributes, respectively.

6.2. Security Analysis

We first provide an intuitive security argument. Recall that a confidential file f is encrypted in the form of $C_f = [U_0, U_{u_1}, \ldots, U_{u_n}, U_{12}, \ldots, U_{1t_1}, U_1, \ldots, U_{N2}, \ldots, U_{Nt_N}, U_N, V] =$

$[rP_0, rP_{u_1}, \ldots, rP_{u_n}, rP_{12}, \ldots, rP_{1t_1}, r\sum_{j=1}^{n_1} P_{a_{1j}}, \ldots, rP_{N2}, \ldots, rP_{Nt_N}, r\sum_{j=1}^{n_N} P_{a_{Nj}}, f \oplus H_2(\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit}))]$. Therefore, the adversary $\mathcal{A}$ needs to construct $\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit})$ to decrypt C_f.

Q_0 and $P_{\diamondsuit}$ can be publicly available, and $n_{\mathbb{A}}$ can be obtained from the ciphertext, but due to the DHP assumption [6], adversary $\mathcal{A}$ is unaware of the value of random mask r. In other words, adversary $\mathcal{A}$ cannot construct $\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit})$ directly. However, due to the properties of the bilinear map, we have: $\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit}) = \hat{e}(U_0, n_{\mathbb{A}}\mathrm{SK}_{\diamondsuit})$. That is to say, adversary $\mathcal{A}$ can construct $\hat{e}(U_0, n_{\mathbb{A}}\mathrm{SK}_{\diamondsuit})$ instead of $\hat{e}(Q_0, rn_{\mathbb{A}}P_{\diamondsuit})$ to decrypt C_f.

To create such a bilinear map, the adversary can only use keys that have been obtained in a security game defined in Section 4, where the constraints are (1) None of the ID $\in \mathbb{R}$ appears in any private key query; (2) None of the users possess a sufficient set of attribute keys to decrypt C_f. For ease of presentation, we have the following assumptions: (1) Adversary $\mathcal{A}$ has requested private key for any $\mathcal{U}_j$ outside $\mathcal{U}_1, \ldots, \mathcal{U}_n$; (2) 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 $\mathrm{SK}_{\diamondsuit}$ is in the user private key and user attribute secret key, so the adversary has to:

(1) Make use of user private key requested for $\mathcal{U}_j$ for the bilinear map, yielding:

$$\begin{aligned}
&\hat{e}(n_{\mathbb{A}}U_0, \mathrm{SK}_{\diamondsuit} + mk_{\diamondsuit}P_{u_j} + \alpha) \\
&= \hat{e}(n_{\mathbb{A}}U_0, \mathrm{SK}_{\diamondsuit})\hat{e}(n_{\mathbb{A}}U_0, mk_{\diamondsuit}P_{u_j}))\hat{e}(n_{\mathbb{A}}U_0, \alpha) \\
&= \hat{e}(U_0, n_{\mathbb{A}}\mathrm{SK}_{\diamondsuit})\hat{e}(mk_{\diamondsuit}P_0, rn_{\mathbb{A}}P_{u_j}))\hat{e}(rn_{\mathbb{A}}P_0, \alpha) \\
&= \hat{e}(U_0, n_{\mathbb{A}}\mathrm{SK}_{\diamondsuit})\hat{e}(Q_{\diamondsuit}, rn_{\mathbb{A}}P_{u_j})\hat{e}(rn_{\mathbb{A}}P_0, \alpha)
\end{aligned}$$

for some α. To obtain $\hat{e}(U_0, n_{\mathbb{A}}\mathrm{SK}_{\diamondsuit})$, the values of $\hat{e}(Q_{\diamondsuit}, rn_{\mathbb{A}}P_{u_j})$ and $\hat{e}(rn_{\mathbb{A}}P_0, \alpha)$ have to be eliminated. However, the value of $\hat{e}(Q_{\diamondsuit}, rn_{\mathbb{A}}P_{u_j})$ is unknown to the adversary, and cannot be constructed by the adversary. Therefore, $\mathcal{A}$ cannot recover the file.

(2) Make use of user attribute secret keys requested for $\mathcal{U}$ and $\mathcal{U}'$ for the bilinear map, yielding:

$$\begin{aligned}
&\hat{e}(U_0, \tfrac{n_{\mathbb{A}}}{n_i}\sum_{j=1, j\neq k}^{n_i} \mathrm{SK}_{it_i,u,a_{ij}} + \tfrac{n_{\mathbb{A}}}{n_i}\mathrm{SK}_{it_i,u',a_{ik}} + \alpha) \\
&= \hat{e}(rP_0, \tfrac{n_{\mathbb{A}}}{n_i}\sum_{j=1, j\neq k}^{n_i}(\mathrm{SK}_{i1} + \sum_{t=2}^{t_i} mk_{i(t-1)}P_{it} + mk_{it_i}mk_uP_{a_{ij}}) \\
&+ \tfrac{n_{\mathbb{A}}}{n_i}\mathrm{SK}_{i1} + \tfrac{n_{\mathbb{A}}}{n_i}\sum_{t=2}^{t_i} mk_{i(t-1)}P_{it} + \tfrac{n_{\mathbb{A}}}{n_i}mk_{it_i}mk_{u'}P_{a_{ik}} + \alpha) \\
&= \hat{e}(rP_0, n_{\mathbb{A}}\mathrm{SK}_{i1})\hat{e}(mk_{it_i}mk_uP_0, \tfrac{n_{\mathbb{A}}}{n_i}r\sum_{j=1, j\neq k}^{n_i} P_{a_{ij}}) \\
&= \hat{e}(U_0, \mathrm{SK}_{\diamondsuit})^{n_{\mathbb{A}}}\prod_{t=2}^{t_i}\hat{e}(Q_{i(t-1)}, U_{it})^{n_{\mathbb{A}}}\hat{e}(rP_0, \alpha) \\
&\quad \hat{e}(mk_{u'}mk_{it_i}P_0, rP_{a_{ik}})^{\frac{n_{\mathbb{A}}}{n_i}}\hat{e}(mk_umk_{it_i}P_0, r\sum_{j=1, j\neq k}^{n_i} P_{a_{ij}})^{\frac{n_{\mathbb{A}}}{n_i}}
\end{aligned}$$

for some α. To obtain $\hat{e}(U_0, \mathrm{SK}_{\diamondsuit})^{n_{\mathbb{A}}}$, 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 $\hat{e}(mk_umk_{it_i}P_0, r\sum_{j=1, j\neq k}^{n_i} P_{a_{ij}})$ are unknown to the adversary, and cannot be constructed. Therefore, $\mathcal{A}$ cannot recover the file.

7. CONCLUSION

Cloud computing is one of the current most important and promising technologies. For the sake of enjoying a more comprehensive and high-quality service, we proposed a novel encryption scheme which simultaneously achieves flexibility, high performance, and full key delegtaion. The proposed scheme, which is collusion resistant, can be proved to have semantic security under the BDH assumption and random oracle model. In future work, we will design a more expressive encryption scheme to have full security under the standard model, with better performance.

REFERENCES

1. L. Vaquero, L. Merino, J. Caceres, and M. Lindner. A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, 39(1): 50-55, January 2009.
2. K. Bowers, A. Juels, and A. Oprea. HAIL: A High-Availability and Integrity Layer for Cloud Storage. In *Proceedings of ACM CCS 2009*, pages 187-198.
3. H. Haclgiimfi, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over Encrypted Data in Database-Service-Provider Model. In *Proceedings of ACM SIGMOD 2002*, pages 216-227.
4. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Proceedings of CRYPTO 1984*, volume 196 of *LNCS*, pages 47-53.
5. C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues, In *Proceedings of IMA 2001*, volume 2260 of *LNCS*, pages 360-363.
6. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of CRYPTO 2001*, volume 2139 of *LNCS*, pages 213-229.
7. J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. In *Proceedings of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466-481.
8. C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. In *Proceedings of ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548-566.
9. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption without Random Oracles. In *Proceedings of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223-238.
10. D. Boneh, X. Boyen, and E. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Proceedings of EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440-456.
11. C. Gentry and S. Halevi. Hierarchical Identity Based Encryption with Polynomially Many Levels. In *Proceedings of TCC 2009*, volume 5444 of *LNCS*, pages 437-456.
12. B. Waters. Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In *Proceedings of CRYPTO 2009*, volume 5677 of *LNCS*, pages 619-636.
13. Q. Liu, G. Wang, and J. Wu. Efficient Sharing of Secure Cloud Storage Services. In *Proceedings of IEEE TSP 2010, in conjunction with IEEE CIT 2010*, pages 922-929.
14. A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *Proceedings of EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457-473.
15. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data. In *Proceedings of ACM CCS 2006*, pages 89-98.
16. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-Based Encryption with Non-Monotonic Access Structures. In *Proceedings of ACM CCS 2007*, pages 195-203.
17. J . Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *Proceedings of ISSP 2007*, pages 321-334.
18. S. Muller, S. Katzenbeisser, and C. Eckert. Distributed Attribute-Based Encryption. In *Proceedings of ICISC 2008*, pages 20-36.
19. M. Chase. Multi-Authority Attribute Based Encryption. In *Proceedings of TCC 2007*, volume 4392 of *LNCS*, pages 515-534.
20. M. Chase and S. Chow. Improving Privacy and Security in Multi-Authority Attribute-Based Encryption. In *Proceedings of ACM CCS 2009*, pages 121-130.

21. S. Yu, C. Wang, K. Ren, and W. Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *Proceedings of INFOCOM 2010*, pages 534-542.
22. G. Wang, Q. Liu, and J. Wu. Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services. In *Proceedings of ACM CCS 2010, P&D Session*, pages 735-737.
23. S. Yu, C. Wang, and K. Ren. Attribute Based Data Sharing with Attribute Revocation. In *Proceeding of ASIACCS 2010*, pages 261-270.
24. E. Goh, H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing Remote Untrusted Storage. In *Proceedings of NDSS 2003*, pages 131-145.
25. M. Blaze, G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *Proceedings of EUROCRYPT 1998*, pages 127–144.
26. A. Fiat and M. Naor. Broadcast Encryption. In *Proceedings of CRYPTO 1993*, volume 773 of *LNCS*, pages 480-491.
27. D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. In *Proceedings of CRYPTO 2002*, volume 2442 of *LNCS*, pages 47-60.
28. D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Proceedings of CRYPTO 2005*, volume 3621 of *LNCS*, pages 258-275.

APPENDIX .

A. SECURITY PROOF

To analyze the security of the proposed scheme, we provide the following theorem, which shows that the PFIBE scheme is semantically secure under the BDH assumption and random oracle model.

Theorem A.1: Suppose that Algorithm $\mathcal{A}$ is an adversary that has the advantage ϵ of successfully attacking the PFIBE scheme. Suppose Algorithm $\mathcal{A}$ specifies a recipient ID set $\mathbb{R} = \{\mathrm{ID}_{u_1}, \ldots, \mathrm{ID}_{u_n}\}$ and a DNF attribute-based access control policy $\mathbb{A} = \overset{N}{\underset{i=1}{\vee}}(CC_i) = \overset{N}{\underset{i=1}{\vee}}(\overset{n_i}{\underset{j=1}{\wedge}} a_{ij})$, and makes at most $q_{H_2} > 0$ hash queries to H_2, at most $q_{E_1} > 0$ user private key queries, and at most $q_{E_2} > 0$ user attribute secret key queries in Dom_A. Then, there is an adversary $\mathcal{B}$ that breaks the BDH problem with the advantage at least $2\epsilon N^N n^n / q_{H_2} e^{N+n} (q_{E_2} + N)^N (q_{E_1} + n)^n$, and a running time $O(time(\mathcal{A}))$. Here, $e \approx 2.71$ is the base of the natural logarithm.

Proof: Let H_1 and H_2 be random oracles from $\{0,1\}^*$ to $\mathbb{G}_1$ and from $\mathbb{G}_2$ to $\{0,1\}^n$, 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 $\mathbb{Z}_q$. Its 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:

Setup: Algorithm $\mathcal{B}$ sets $Q_0 = \mu_0 = \alpha P_0$ and gives $< q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, Q_0, H_1, H_2 >$ as the system parameters to Algorithm $\mathcal{A}$, where H_1 and H_2 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 $(\text{ID-tuple}_{a_j}, \text{P-tuple}_{a_j}, \text{b-tuple}_{a_j}, \text{mk-tuple}_{a_j}, \text{c-tuple}_{a_j})$, or $(\text{ID-tuple}_{u_j}, \text{P-tuple}_{u_j}, \text{b-tuple}_{u_j}, \text{mk-tuple}_{u_j}, \text{c-tuple}_{u_j})$. H_1-List is initially empty. Algorithm $\mathcal{B}$ first adds $\text{ID-tuple}_{i1} = (\mathrm{ID}_{\diamondsuit})$ to H_1-List as follows:

1. Pick random elements $mk_{\diamondsuit}$ and $b_{\diamondsuit} \in \mathbb{Z}_q$.
2. Set $c_{\diamondsuit} = 1$.
3. Set $P_{\diamondsuit} = b_{\diamondsuit} P_0 + \mu_1$.
4. Put $((\text{ID}_{\diamondsuit}), (P_{\diamondsuit}), (b_{\diamondsuit}), (mk_{\diamondsuit}), (c_{\diamondsuit}))$ in H$_1$-List.

When Algorithm $\mathcal{A}$ queries H_1 at a point ID-tuple$_{u_i}$ $= (\text{PK}_{\diamondsuit}, \text{ID}_{u_i}) = (\text{ID}_{\diamondsuit}, \text{ID}_{u_i})$, Algorithm $\mathcal{B}$ responds as follows:

1. Picks two random elements mk_{u_i} and $b_{u_i} \in \mathbb{Z}_q$.
2. Picks a random coin $c_{u_i} \in \{0,1\}$ so that $Pr[c_{u_i} = 0] = \delta_1$ for some δ_1 that will be determined later.
3. If $c_{u_i} = 1$, Algorithm $\mathcal{B}$ sets $P_{u_i} = b_{u_i} P_0$.
4. If $c_{u_i} = 0$, Algorithm $\mathcal{B}$ sets $P_{u_i} = b_{u_i} P_0 - mk_{\diamondsuit}^{-1} P_{\diamondsuit}$, where $mk_{\diamondsuit}^{-1}$ is the inverse of $mk_{\diamondsuit}$ modulo q.
5. Put $((\text{ID}_{\diamondsuit}, \text{ID}_{u_i}), (P_{\diamondsuit}, P_{u_i}), (b_{\diamondsuit}, b_{u_i}), (mk_{\diamondsuit}, mk_{u_i}), (c_{\diamondsuit}, c_{u_i}))$ in H$_1$-List and return $H_1(\text{ID-tuple}_{u_i}) = (P_{\diamondsuit}, P_{u_i}) \in \mathbb{G}_1$ to Algorithm $\mathcal{A}$.

When Algorithm $\mathcal{A}$ queries H_1 at a point ID-tuple$_{a_i}$ $= (\text{ID}_{i1}, \ldots, \text{ID}_{it_i}, \text{ID}_{a_i})$, Algorithm $\mathcal{B}$ responds as follows: Let y be maximal such that $(\text{ID}_{i1}, \ldots, \text{ID}_{iy}) = (\text{ID}_{j1}, \ldots, \text{ID}_{jy})$ for some tuple $((\text{ID}_{j1}, \ldots, \text{ID}_{jt_j}, \text{ID}_{a_j}), (P_{j1}, \ldots, P_{jt_j}, P_{a_j}), (b_{j1}, \ldots, b_{jt_j}, b_{a_j}), (mk_{j1}, \ldots, mk_{jt_j}, mk_{a_j}), (c_{j1}, \ldots, c_{jt_j}, c_{a_j}))$ 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 $c_{ik} = 1$.
   (c) Sets $P_{ik} = b_{ik} P_0$.
3. For ID$_{a_i}$ Algorithm $\mathcal{B}$:
   (a) Picks two random elements mk_{a_i} and $b_{a_i} \in \mathbb{Z}_q$.
   (b) Picks a random coin $c_{a_i} \in \{0,1\}$ so that $Pr[c_{a_i} = 0] = \delta_2$ for some δ_2 that will be determined later.
   (c) If $c_{a_i} = 1$, Algorithm $\mathcal{B}$ sets $P_{a_i} = b_{a_i} P_0$.
   (d) If $c_{a_i} = 0$, Algorithm $\mathcal{B}$ sets $P_{a_i} = b_{a_i} P_0 - mk_{it_i}^{-1} P_{\diamondsuit}$, where $mk_{it_i}^{-1}$ is the inverse of mk_{it_i} modulo q.
4. Put $((\text{ID}_{i1}, \ldots, \text{ID}_{it_i}, \text{ID}_{a_i}), (P_{i1}, \ldots, P_{it_i}, P_{a_i}), (b_{i1}, \ldots, b_{it_i}, b_{a_i}), (mk_{i1}, \ldots, mk_{it_i}, mk_{a_i}), (c_{i1}, \ldots, c_{it_i}, c_{a_i}))$ in H$_1$-List and return $H_1(\text{ID-tuple}_{a_i}) = (P_{i1}, \ldots, P_{it_i}, P_{a_i}) \in \mathbb{G}_1$ to Algorithm $\mathcal{A}$.

Note that these values are always chosen uniformly in $\mathbb{G}_1$, and are independent of Algorithm $\mathcal{A}$'s view as required.

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 $H_2(T_i) = V_i$.

Phase 1: At any time, Algorithm $\mathcal{A}$ may make a private key query on any ID-tuple$_{u_i}$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the H$_1$-Queries to obtain the appropriate tuple (ID-tuple$_{u_i}$, P-tuple$_{u_i}$, b-tuple$_{u_i}$, mk-tuple$_{u_i}$, c-tuple$_{u_i}$) in H$_1$-List.
2. If $c_{u_i} = 1$, then Algorithm $\mathcal{B}$ reports failure to Algorithm $\mathcal{A}$ and terminates the interaction. Otherwise, we know that $P_{u_i} = b_{u_i}P_0 - mk_{\Diamond}^{-1}P_{\Diamond}$ where $mk_{\Diamond}^{-1}$ is the inverse of $mk_{\Diamond}$ modulo q.
3. Set Q-tuple$_{\Diamond} = (Q_0, Q_{\Diamond})$, where $Q_0 = \mu_0 = \alpha P_0$, and $Q_{\Diamond} = mk_{\Diamond}\mu_0 = mk_{\Diamond}\alpha P_0$.
4. Set SK_{u_i} = (Q-tuple$_{\Diamond}$, $mk_{\Diamond} b_{u_i}\mu_0$) $= (mk_{\Diamond}\alpha P_0, mk_{\Diamond} b_{u_i}\alpha P_0)$ as the private key to Algorithm $\mathcal{A}$.

Observe that: $Q_0 = \mu_0 = \alpha P_0$, $Q_{\Diamond} = \alpha mk_{\Diamond}P_0$, so the root master key is α, and the master key of DM$_{\Diamond}$ is $\alpha mk_{\Diamond}$. Although Algorithm $\mathcal{B}$ does not know the values of α, it can output a correct private key for ID-tuple$_{u_i}$ as follows:

By definition, the value of SK_{u_i} should be $\mathrm{SK}'_{\Diamond} + mk'_{\Diamond}P_{u_i}$, where symbol $'$ denotes authentical keys. Observe that:

$$\begin{aligned}
\mathrm{SK}_{u_i} &= \mathrm{SK}'_{\Diamond} + mk'_{\Diamond}P_{u_i} \\
&= \alpha P_{\Diamond} + \alpha mk_{\Diamond}(b_{u_i}P_0 - mk_{\Diamond}{}^{-1}P_{\Diamond}) \\
&= \alpha P_{\Diamond} + \alpha mk_{\Diamond} b_{u_i}P_0 - \alpha mk_{\Diamond} mk_{\Diamond}{}^{-1}P_{\Diamond} \\
&= \alpha P_{\Diamond} + \alpha mk_{\Diamond} b_{u_i}P_0 - \alpha P_{\Diamond} \\
&= \alpha mk_{\Diamond} b_{u_i}P_0
\end{aligned}$$

as required.

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

1. Run the H$_1$-Queries to obtain the appropriate tuple (ID-tuple$_{u_i}$, P-tuple$_{u_i}$, b-tuple$_{u_i}$, mk-tuple$_{u_i}$, c-tuple$_{u_i}$) in H$_1$-List.
2. Run the H$_1$-Queries to obtain the appropriate tuple (ID-tuple$_{a_j}$, P-tuple$_{a_j}$, b-tuple$_{a_j}$, mk-tuple$_{a_j}$, c-tuple$_{a_j}$) in H$_1$-List.
3. If $c_{a_j} = 1$, then Algorithm $\mathcal{B}$ reports failure to Algorithm $\mathcal{A}$ and terminates the interaction. Otherwise, we know that $P_{a_j} = b_{a_j}P_0 - mk_{jt_j}^{-1}P_{\Diamond}$ where $mk_{jt_j}^{-1}$ is the inverse of mk_{jt_j} modulo q.
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 $\mathrm{SK}_{jt_j,u_i,a_j} = \sum_{k=2}^{t_j} mk_{j(k-1)}P_{jk} + mk_{jt_j}mk_{u_i}P_{a_j} + mk_{jt_j}b_{a_j}\mu_0$.
6. Return $(\mathrm{SK}_{jt_j,u_i}, \mathrm{SK}_{jt_j,u_i,a_j})$ to $\mathcal{A}$.

Observe that: $Q_0 = \mu_0 = \alpha P_0$, $Q_{jk} = mk_{jk}P_0$ for $1 \le k \le t_j - 1$, and SK_{jt_j,u_i} = (Q-tuple$_{j(t_j-1)}$, $mk_{jt_j}(mk_{u_i} + \alpha)P_0$), so the root master key is α, the master key of DM$_{jk}$

is mk_{jk} where $1 \le k \le t_j$, and the user master key is $mk_{u_i} + \alpha$. 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 $\mathrm{SK}_{jt_j,u_i,a_j}$ should be: $\mathrm{SK}'_{j1} + \sum_{k=2}^{t_j} mk'_{j(k-1)} P_{jk} + mk'_{jt_j} mk'_{u_i} P_{a_j}$, where symbol $'$ denotes authentical keys. Observe that:

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

as required.

Challenge: Once Algorithm $\mathcal{A}$ decides that Phase 1 is over, it outputs a recipient ID set $\mathbb{R} = \{\mathrm{ID}_1, \ldots, \mathrm{ID}_n\}$, a DNF attribute-based access policy $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij})$ and two plaintext f_0, f_1 on which it wishes to be challenged. Algorithm $\mathcal{B}$ responds as follows:

1. Run the H$_1$-Queries to obtain the appropriate tuples (ID-tuple$_{u_i}$, P-tuple$_{u_i}$, b-tuple$_{u_i}$, mk-tuple$_{u_i}$, c-tuple$_{u_i}$) in H$_1$-List, for $1 \le i \le n$.
2. If $c_{u_i} = 0$, then Algorithm $\mathcal{B}$ reports failure to Algorithm $\mathcal{A}$ and terminates the interaction. Otherwise, we know that $P_{u_i} = b_{u_i} P_0$, for $1 \le i \le n$.
3. For $1 \le i \le N$: Run the H$_1$-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$_1$-List.
4. 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 $a_{i\spadesuit}$ with $P_{a_{i\spadesuit}} = mk_{it_i} b_{a_{i\spadesuit}} P_0$ where $1 \le i \le N$. Therefore, there is at least one attribute secret key in either of conjunctive clauses hasn't been queried by Algorithm $\mathcal{A}$ in Phase 1.
5. Pick a random $b \in \{0,1\}$ and a random string $J \in \{0,1\}^n$, and give the ciphertext CT= $(\mathbb{A}, [U_0, U_{u_1}, \ldots, U_{u_n}, U_{12}, \ldots, U_{1t_1}, U_1, \ldots, U_{N2}, \ldots, U_{Nt_N}, U_N, V]) = (\mathbb{A}, [\mu_2, b_{u_1}\mu_2, \ldots, b_{u_n}\mu_2, b_{12}\mu_2, \ldots, b_{1t_1}\mu_2, b_{a_{1\spadesuit}}\mu_2, \ldots, b_{N2}\mu_2, \ldots, b_{Nt_N}\mu_2, b_{a_{N\spadesuit}}\mu_2, J])$.

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

$$\begin{aligned} J &= f_b \oplus H_2(\hat{e}(\gamma\alpha P_0, b_\diamondsuit P_0 + \beta P_0)) \\ &= f_b \oplus H_2(\hat{e}(P_0, P_0)^{\alpha\gamma(\beta + b_\diamondsuit)}). \end{aligned}$$

By definition, user $\mathcal{U}$, whose ID belonging to $\mathbb{R}$ or all attribute satisfying CC_i in $\mathbb{A}$, can decrypt the ciphertext using the *RDecrypt* algorithm under $\mathrm{SK}_u^{'}$, or using the *ADecrypt* algorithm under $\mathrm{SK}_{it_i,u}^{'}$ and $\mathrm{SK}_{it_i,u,a_{i\spadesuit}}^{'}$, where symbol $'$ denotes authentic keys.

He computes $J \oplus H_2(\hat{e}(U_0, \mathrm{SK}_\diamondsuit^{'} + mk_\diamondsuit^{'} P_{u_i})/\hat{e}(Q_\diamondsuit^{'}, U_{u_i}))$ to recover the file f_b. Observe that:

$$\begin{aligned} & J \oplus H_2(\hat{e}(U_0, \mathrm{SK}_\diamondsuit^{'} + mk_\diamondsuit^{'} P_{u_i})/\hat{e}(Q_\diamondsuit^{'}, U_{u_i})) \\ &= J \oplus H_2(\hat{e}(\mu_2, \alpha P_\diamondsuit + mk_\diamondsuit \alpha P_{u_i})/\hat{e}(Q_\diamondsuit, b_{u_i}\mu_2)) \\ &= J \oplus H_2(\hat{e}(\mu_2, \alpha P_\diamondsuit)\hat{e}(\mu_2, mk_\diamondsuit \alpha P_{u_i})/\hat{e}(Q_\diamondsuit, b_{u_i}\mu_2)) \\ &= J \oplus H_2(\hat{e}(\mu_2, \alpha P_\diamondsuit)\hat{e}(\gamma P_0, mk_\diamondsuit \alpha P_{u_i})/\hat{e}(Q_\diamondsuit, b_{u_i}\mu_2)) \\ &= J \oplus H_2(\hat{e}(\mu_2, \alpha P_\diamondsuit)\hat{e}(mk_\diamondsuit \alpha P_0, \gamma P_{u_i})/\hat{e}(Q_\diamondsuit, b_{u_i}\mu_2)) \\ &= J \oplus H_2(\hat{e}(\gamma P_0, \alpha P_\diamondsuit)\hat{e}(Q_\diamondsuit, b_{u_i}\gamma P_0)/\hat{e}(Q_\diamondsuit, b_{u_i}\gamma P_0)) \\ &= J \oplus H_2(\hat{e}(\gamma\alpha P_0, P_\diamondsuit)) \\ &= J \oplus H_2(\hat{e}(\gamma\mu_0, P_\diamondsuit)) = f_b \end{aligned}$$

He computes $J \oplus H_2(\frac{\hat{e}(U_0, \mathrm{SK}_{it_i,u,a_{i\spadesuit}}^{'})}{\hat{e}(mk_{it_i}^{'} mk_u^{'} P_0, U_i) \prod_{k=2}^{t_i} \hat{e}(U_{ik}, Q_{i(k-1)}^{'})})$ to recover the file f_b. Observe that:

$$\begin{aligned} & J \oplus H_2(\frac{\hat{e}(U_0, \mathrm{SK}_{it_i,u,a_{i\spadesuit}}^{'})}{\hat{e}(mk_{it_i}^{'} mk_u^{'} P_0, U_i) \prod_{k=2}^{t_i} \hat{e}(U_{ik}, Q_{i(k-1)}^{'})}) \\ &= J \oplus H_2(\frac{\hat{e}(U_0, \mathrm{SK}_{i1}^{'} + \sum_{k=2}^{t_i} mk_{i(k-1)} P_{ik} + mk_{it_i}(mk_u + \alpha) P_{a_{i\spadesuit}})}{\hat{e}(mk_{it_i}(mk_u + \alpha) P_0, b_{a_{i\spadesuit}} \mu_2) \prod_{k=2}^{t_i} \hat{e}(U_{ik}, Q_{i(k-1)}^{'})}) \\ &= J \oplus H_2(\frac{\hat{e}(\gamma P_0, \alpha P_1) \prod_{k=2}^{t_i} \hat{e}(U_{ik}, Q_{i(k-1)}^{'}) e(\gamma P_0, mk_{it_i}(mk_u + \alpha) P_{a_{i\spadesuit}})}{\hat{e}(mk_{it_i}(mk_u + \alpha) P_0, b_{a_{i\spadesuit}} \gamma P_0) \prod_{k=2}^{t_i} \hat{e}(U_{i_k}, Q_{i(k-1)}^{'})}) \\ &= J \oplus H_2(\frac{\hat{e}(\gamma P_0, \alpha P_1) e(\gamma P_0, mk_{it_i}(mk_u + \alpha) P_{a_{i\spadesuit}})}{\hat{e}(\gamma P_0, mk_{it_i}(mk_u + \alpha) P_{a_{i\spadesuit}})}) \\ &= J \oplus H_2(\hat{e}(\gamma P_0, \alpha P_{i1}) \\ &= J \oplus H_2(\hat{e}(\gamma\mu_0, P_\diamondsuit) = f_b \end{aligned}$$

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

Phase 2. Algorithm $\mathcal{A}$ can continue issuing more private key queries other than ID-tuple$_{u_i}$, ..., ID-tuple$_{u_n}$, and attribute secret key queries other than ID-tuple$_{a_{1\spadesuit}}$, ..., ID-tuple$_{a_{N\spadesuit}}$. 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 $T_i/\hat{e}(\mu_0, \mu_2)^{b_\diamondsuit}$ as the solution to D.

To complete the proof of *Theorem A.1*, we now show that Algorithm $\mathcal{B}$ correctly outputs D with the probability at least $2\epsilon N^N n^n / q_{H_2} e^{N+n} (q_{E_2} + N)^N (q_{E_1} + n)^n$. 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_1} private key queries and q_{E_2} attribute secret key queries. Then, the probability that Algorithm $\mathcal{B}$ does not abort in Phase 1 or 2 is $\delta_1^{q_{E_1}}\delta_2^{q_{E_2}}$. And the probability that it does not abort during the challenge step is $(1-\delta_1)^n(1-\delta_2)^N$. Therefore, the probability that Algorithm $\mathcal{B}$ does not abort during the simulation is $\delta_1^{q_{E_1}}\delta_2^{q_{E_2}}(1-\delta_1)^n(1-\delta_2)^N$. These values are maximized at $\delta_1^{opt} = 1-n/(q_{E_1}+n)$, and $\delta_2^{opt} = 1-N/(q_{E_2}+N)$. Using δ_1^{opt} and δ_2^{opt}, the probability that Algorithm $\mathcal{B}$ does not abort is at least $(N/e(q_{E_2}+N))^N(n/e(q_{E_1}+n))^n$.

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 $\neg Q$, 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 $\mathcal{A}$ outputs the correct result, therefore, in the real attack $Pr[b = b^{'}|\neg Q] = 1/2$. Since Algorithm $\mathcal{A}$ has the advantage ϵ, $|Pr[b = b^{'}|\neg Q] - 1/2| \geq \epsilon$. According to the following formulae, we know $Pr[Q] \geq 2\epsilon$.

$$\begin{array}{rcl}
Pr[b = b^{'}] & = & Pr[b = b^{'}|\neg Q]Pr[\neg Q] \\
 & + & Pr[b = b^{'}|Q]Pr[Q] \\
 & \leq & 1/2Pr[\neg Q] + Pr[Q] \\
 & = & 1/2 + 1/2Pr[Q] \\
Pr[b = b^{'}] & \geq & Pr[b = b^{'}|\neg Q]Pr[\neg Q] \\
 & = & 1/2Pr[\neg Q] \\
 & = & 1/2 - 1/2Pr[Q]
\end{array}$$

Therefore, we have that $Pr[Q] \geq 2\epsilon$ 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 $(N/e(q_{E_2}+N))^N(n/e(q_{E_1}+n))^n$, we see that Algorithm $\mathcal{B}$'s success probability is at least $\epsilon' = 2\epsilon N^N n^n/q_{H_2}e^{N+n}(q_{E_2}+N)^N(q_{E_1}+n)^n$. ■