Protecting Your Right: Attribute-based Keyword Search with Fine-grained Owner-enforced Search Authorization in the Cloud

Wenhai Sun*†, Shucheng Yu‡, Wenjing Lou†, Y. Thomas Hou†, and Hui Li*
*The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi, China
†Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
‡University of Arkansas at Little Rock, Little Rock, AR, USA

***Abstract*—Search over encrypted data is a critically important enabling technique in cloud computing, where encryption-before-outsourcing is a fundamental solution to protecting user data privacy in the untrusted cloud server environment. Many secure search schemes have been focusing on the single-contributor scenario, where the outsourced dataset or the secure searchable index of the dataset are encrypted and managed by a single owner, typically based on symmetric cryptography. In this paper, we focus on a different yet more challenging scenario where the outsourced dataset can be contributed from multiple owners and are searchable by multiple users, i.e. multi-user multi-contributor case. Inspired by attribute-based encryption (ABE), we present the first attribute-based keyword search scheme with efficient user revocation (ABKS-UR) that enables scalable fine-grained (i.e. file-level) search authorization. Our scheme allows multiple owners to encrypt and outsource their data to the cloud server independently. Users can generate their own search capabilities without relying on an always online trusted authority. Fine-grained search authorization is also implemented by the owner-enforced access policy on the index of each file. Further, by incorporating proxy re-encryption and lazy re-encryption techniques, we are able to delegate heavy system update workload during user revocation to the resourceful semi-trusted cloud server. We formalize the security definition and prove the proposed ABKS-UR scheme selectively secure against chosen-keyword attack. Finally, performance evaluation shows the efficiency of our scheme.**

I. Introduction

Cloud computing has emerged as a new enterprise IT architecture. Many companies are moving their applications and databases into the cloud and start to enjoy many unparalleled advantages brought by cloud computing, such as on-demand computing resource configuration, ubiquitous and flexible access, considerable capital expenditure savings, etc. However, privacy concern has remained a primary barrier preventing the adoption of cloud computing by a broader range of users/applications. When sensitive data are outsourced to the cloud, data owners naturally become concerned with the privacy of their data in the cloud and beyond. Encryption-before-outsourcing has been regarded as a fundamental means of protecting user data privacy against the cloud server [1], [2], [3]. However, how the encrypted data can be effectively utilized then becomes another new challenge. Significant attention has been given and much effort has been made to address this issue, from secure search over encrypted data [4], secure function evaluation [5], to fully homeomorphic encryption systems [6] that provide generic solution to the problem in theory but are still too far from being practical due to the extremely high complexity.

978-1-4799-3360-0/14/$31.00

This paper focuses on the problem of search over encrypted data, which is an important enabling technique for the encryption-before-outsourcing privacy protection paradigm in cloud computing, or in general in any networked information system where servers are not fully trusted. Much work has been done, with majority focusing on the single-contributor scenario, i.e. the dataset to be searched is encrypted and managed by a single entity, which we call *owner* or *contributor* in this paper. Under this setting, to enable search over encrypted data, the owner has to either share the secret key with authorized users [4], [7], [8], or stay online to generate the search *trapdoors*, i.e. the "encrypted" form of keywords to be searched, for the users upon request [9], [10]. The same symmetric key will be used to encrypt the dataset (or the searchable index of the dataset) and to generate the trapdoors. These schemes seriously limit the users' search flexibility.

Consider a file sharing system that hosts a large number of files, contributed from multiple owners and to be shared among multiple users (e.g. 4shared.com, mymedwall.com). This is a more challenging multi-owner multi-user scenario. How to enable multiple owners to encrypt and add their data to the system and make it searchable by other users? Moreover, data owners may desire fine-grained search authorization that only allows their authorized users to search their contributed data. By *fine-grained*, we mean the search authorization is controlled at the granularity of per file level. Symmetric cryptography based schemes [4], [7], [8] are clearly not suitable for this setting due to the high complexity of secret key management. Although authorized keyword search can be realized in single-owner setting by explicitly defining a server-enforced user list that takes the responsibility to control legitimate users' search capabilities [11], [12], i.e. search can only be carried out by the server with the assistance of legitimate users' complementary keys on the user list, these schemes did not

realize fine-grained owner-enforced search authorization and thus are unable to provide differentiated access privileges for different users within a dataset. Asymmetric cryptography is better suited to this dynamic setting by encrypting individual contribution with different public keys. For example, Hwang et al. [13] implicitly defined a user list for each file by encrypting the index of the file with all the public keys of the intended users. However, extending such user list approach to the multi-owner setting and on a per file basis is not trivial as it would impose significant scalability issue considering a potential large number of users and files supported by the system. Additional challenges include how to handle the updates of the user lists in the case of user enrollment, revocation, etc., under the dynamic cloud environment.

In this paper, we address these open issues and present an authorized keyword search scheme over encrypted cloud data with efficient user revocation in the multi-user multi-data-contributor scenario. We realize *fine-grained owner-enforced search authorization* by exploiting ciphertext policy attribute-based encryption (CP-ABE) technique. Specifically, the data owner encrypts the index of each file with an access policy created by him/her, which defines what type of users can search this index. The data user generates the trapdoor independently without relying on an always online trusted authority (TA). The cloud server (CS) can search over the encrypted indexes with the trapdoor on a user's behalf, and then returns matching results if and only if the user's attributes associated with the trapdoor satisfy the access policies embedded in the encrypted indexes. We differentiate *attributes* and *keywords* in our design. Keywords are actual content of the files while attributes refer to the properties of users. The system only maintains a limited number of attributes for search authorization purpose. Data owners create the index consisting of all keywords in the file but encrypt the index with an access structure only based on the attributes of authorized users, which makes the proposed scheme more scalable and suitable for the large scale file sharing system. In order to further release the data owner from the burdensome user membership management, we use proxy re-encryption [14] and lazy re-encryption [15] techniques to shift the workload as much as possible to the CS, by which our proposed scheme enjoys efficient user revocation. Formal security analysis shows that the proposed scheme is provably secure and meets various search privacy requirements. Performance evaluation also demonstrates its efficiency and practicality.

Our contributions can be summarized as follows:

1) We design a novel and scalable authorized keyword search over encrypted data scheme supporting multiple data users and multiple data contributors. Compared with existing works, our scheme supports fine-grained owner-enforced search authorization at the file level with better scalability for large scale system in that the search complexity is linear to the number of attributes in the system, instead of the number of authorized users.
2) Data owner can delegate most of computationally intensive tasks to the CS, which makes the user revocation process efficient and is more suitable for cloud outsourcing model.
3) We formally prove our proposed scheme selectively secure against chosen-keyword attack.

II. Related Work

A. Keyword Search over Encrypted Data

1) Secret key vs. Public key: Encrypted data search has been studied extensively in the literature. Song et al. [4] designed the first searchable encryption scheme to enable a full text search over encrypted files. Since this seminal work, many secure search schemes have been proposed to boost the efficiency and enrich the search functionalities based on either secret-key cryptography (SKC) [7], [8], [9], [10] or public-key cryptography (PKC) [16], [17], [18]. Curtmola et al. [7] presented an efficient single keyword encrypted data search scheme by adopting inverted index structure. The authors in [8] designed a dynamic version of [7] with the ability to add and delete files efficiently. To enrich search functionalities, Cao et al. [9] proposed the first privacy-preserving multi-keyword ranked search scheme over encrypted cloud data using "coordinate matching" similarity measure. Later on, Sun et al. [10] presented a secure multi-keyword text search scheme in the cloud enjoying more accurate search results by "cosine similarity measure" in the vector space model and practically efficient search process using a tree-based secure index structure. Compared with symmetric search techniques, PKC-based search schemes are able to generate more flexible and more expressive search queries. In [16], Boneh et al. devised the first PKC-based encrypted data search scheme supporting single keyword query. The scheme from [17] supports search queries with conjunctive keywords by explicitly indicating the number of encrypted keywords in an index. Predicate encryption [18], [19] is another promising technique to fulfill the expressive secure search functionality. For example, the proposed scheme in [18] supports conjunctive, subset, and range queries, and disjunctions, polynomial equations, and inner products could be realized in [19].

2) Authorized keyword search: To grant multiple users the search capabilities, user authorization should be enforced. In [11], [12], the authors adopt a server-enforced user list containing all the legitimate users' complementary keys that are used to help complete the search in the enterprise scenario to realize search authorization. But these SKC-based schemes only allow one data contributor in the system. Hwang et al. [13] in the public-key setting presented a conjunctive keyword search scheme in multi-user multi-owner scenario. But this scheme is not scalable under the dynamic cloud environment because the size of the encrypted index and the search complexity is proportional to the number of the authorized users, and to add a new user, the data owner has to rewrite all the corresponding indexes. By exploiting hierarchical predicate encryption, Li et al. [20] proposed a file-level authorized private keyword search (APKS) scheme over encrypted cloud data. However, it incurs additional communication cost, since

whenever users want to search, they have to resort to the attribute authority to acquire the search capabilities. Moreover, this scheme is more suitable for the structured database that contains only limited number of keywords. The search time there is proportional to the total number of keywords in the system, which would be inefficient for arbitrarily-structured data search, e.g., free text search, in the case of dynamic file sharing system.

B. Attribute-based Encryption

There has been a great interest in developing attribute-based encryption [21], [22], [23], [24] due to its fine-grained access control property. Goyal et al. [21] designed the first key policy attribute-based encryption (KP-ABE) scheme, where ciphertext can be decrypted only if the attributes that are used for encryption satisfy the access structure on the user private key. Under the reverse situation, CP-ABE allows user private key to be associated with a set of attributes and ciphertext associated with an access structure. CP-ABE is a preferred choice when designing an access control mechanism in a broadcast environment. Since the first construction of CP-ABE [22], many works have been proposed for more expressive, flexible and practical versions of this technique. Cheung et al. [23] proposed a selectively secure CP-ABE construction in the standard model using the simple boolean function, i.e. AND gate. By adopting proxy re-encryption and lazy re-encryption techniques, Yu et al. [24] also devised a selectively secure CP-ABE scheme with the ability of attribute revocation, which is perfectly suitable for the data-outsourced cloud model.

III. Problem Formulation

A. System Model

The system framework of our proposed ABKS-UR scheme involves three entities: *cloud server*, many *data owners*, and many *data users*, as shown in Fig. 1. In addition, a trusted authority is implicitly assumed to be in charge of generating and distributing public keys, private keys and re-encryption keys. To enforce fine-grained authorized keyword search, the data owner generates the secure indexes with attribute-based access policies before outsourcing them along with the encrypted data into the CS. Note that we can encrypt data by any secure encryption technique, such as AES, which is outside the scope of this paper. To search the datasets contributed from various data owners, a data user generates a trapdoor of keyword of interest using his/her private key and submits it to the CS. We adopt the *per-dataset* user list to enforce the coarse-grained *dataset search authorization*. Thus, our scheme benefits from search process acceleration as search does not need to go to a particular dataset if the user is not on the corresponding user list. Notably, even with the per-dataset user list in place, the enforcement of the search authorization is still controlled by the owner-defined access policy, i.e. the CS will search the corresponding datasets and return the valid search results to the user if and only if the attributes of the user on the trapdoor satisfy the access policies of the secure indexes of the returned files, and the intended keyword is found in these files.

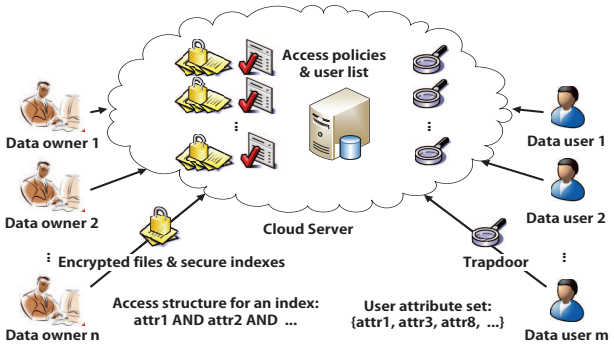


Fig. 1. Framework of authorized keyword search over encrypted cloud data.

B. Threat Model

We consider the CS honest-but-curious, which is also employed by related works on secure search over encrypted data [9], [10], [20]. We assume that the CS honestly follows the designated protocol, but curiously infers additional privacy information based on the data available to him. Furthermore, malicious data users may collude to access files beyond their access privileges by using their secret keys. Analogue to [24], as we delegate most of the system update workload to the CS, we assume that the CS will not collude with the revoked malicious users to help them gain unauthorized access privileges.

C. Design Goals

Our proposed ABKS-UR scheme in the cloud aims to achieve the following functions and security goals:

Authorized Keyword Search: The secure search system should enable data-owner-enforced search authorization, i.e. only users that meet the owner-defined access policy can obtain the valid search results. Besides achieving fine-grained authorization, another challenge is to make the scheme scalable for dynamic cloud environment.

Supporting Multiple Data Contributors and Data Users: The designed scheme should accommodate many data contributors and data users. Each user is able to search over the encrypted data contributed from multiple data owners.

Efficient User Revocation: Another important design goal is to efficiently revoke users from the current system while minimizing the impact on the remaining legitimate users.

Security Goals: In this paper, we are mainly concerned with secure search related privacy requirements, and define them as follows. 1) *Keyword semantic security:* as a novel attribute-based keyword search technique, we will formally prove our proposed ABKS-UR scheme is *semantically secure* against *chosen keyword attack* under *selective ciphertext policy model* (IND-sCP-CKA). The related security definition and semantic security game used in the proof are presented in Appendix.A. 2) *Trapdoor unlinkability:* this security property makes the CS unable to visually distinguish two or more trapdoors even

containing the same keyword. Note that we do not intend to protect *predicate privacy* as the attacker may collude with malicious users or use public key to generate valid secure indexes so as to infer the keyword in a trapdoor from legitimate users, and this privacy breach cannot be protected inherently for any public key based encrypted data search scheme [25]. Moreover, we do not aim to protect *access pattern* in this paper due to the extremely high complexity, i.e. to protect it, algorithm has to "touch" the whole dataset [26].

IV. The Proposed Authorized Keyword Search

We exploit the CP-ABE [23], [24] technique to achieve scalable fine-grained authorized keyword search over encrypted cloud data supporting multiple data owners and data users. Specifically, for each file, the data owner generates an access-policy-protected secure index, where the access structure is expressed as a series of AND gates. Only authorized users with attributes satisfying the access policies can obtain matching results. Otherwise, they have no means to tell whether the search failure comes from a keyword mismatch or an authorization failure. Moreover, we should consider user membership management carefully in the multi-user setting. A naïve solution is to impose the burden on each data owner. As a result, data owner is required to be always online to promptly respond the membership update request, which is impractical and inefficient. By using proxy re-encryption [14], the data owner can delegate most of the workload to the cloud without infringing search privacy.

A. Algorithm Definition

Let $\mathcal{N}$ denote the universal attribute set $\{1, ..., n\}$ for some nature number n. We refer to attributes i and their negations $\neg i$ as literals. $\mathcal{I} \subseteq \mathcal{N}$ is the attribute set used for access structure on encrypted index and here we consider a series of AND gates $\bigwedge_{i \in \mathcal{I}} \underline{i}$ (literal $\underline{i}$ is either positive i or negative $\neg i$). $\mathcal{S} \subseteq \mathcal{N}$ is the attribute set for user secret key.

Definition 1: An attribute-based keyword search with efficient user revocation scheme for keyword space $\mathcal{W}$ and access structure space $\mathcal{G}$ consists of nine fundamental algorithms as follows:

- $\mathsf{Setup}(\lambda, \mathcal{N}) \rightarrow (PK, MK)$: The setup algorithm takes as input the security parameter λ and an attribute universe description $\mathcal{N}$. It defines a bilinear group $\mathbb{G}$ of prime order p with a generator g. Thus, a bilinear map is defined as $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, which has the properties of *bilinearity*, *computability* and *non-degeneracy*. It outputs the public parameters PK and the master secret key MK. The version number ver is initialized as 1.
- $\mathsf{CreateUL}(PK, ID) \rightarrow UL$: The user list generation algorithm takes as input the public parameters PK and the user identity ID. It outputs the user list UL for a dataset.
- $\mathsf{EncIndex}(PK, GT, w) \rightarrow D$: The index encryption algorithm takes as input the current public parameters PK, the access structure $GT \in \mathcal{G}$, a keyword $w \in \mathcal{W}$ and outputs the encrypted index D.
- $\mathsf{KeyGen}(PK, MK, \mathcal{S}) \rightarrow SK$: The key generation algorithm takes as input the current public parameters PK, the current master secret key MK, and the attribute set $\mathcal{S}$ associated with a particular user. It outputs the user's secret key SK.
- $\mathsf{ReKeyGen}(\Phi, MK) \rightarrow (rk, MK', PK')$: The re-encryption key generation algorithm takes as input the attribute set Φ that contains the attributes to be updated, and the current system master key MK. It outputs a set of proxy re-encryption keys rk for all the attributes in $\mathcal{N}$, the updated MK' and PK', where all the version numbers are increased by 1. For the attributes not in Φ, set their proxy re-encryption keys as 1 in rk.
- $\mathsf{ReEncIndex}(\Delta, rk, D) \rightarrow D'$: It takes as input an encrypted index D, the re-encryption key set rk and the attribute set Δ that includes all the attributes in D's access structure with the re-encryption keys not being 1 in rk. Then it outputs a new re-encrypted index D'.
- $\mathsf{ReKey}(\Omega, rk, PSK) \rightarrow PSK'$: It takes as input a user's partial secret key PSK, the re-encryption key set rk and the attribute set Ω that contains all the attributes in PSK with the re-encryption keys not being 1 in rk. Finally, it outputs a new PSK' for that user.
- $\mathsf{GenTrapdoor}(PK, SK, w') \rightarrow Q$: The trapdoor generation algorithm takes as input the current public key PK, the user's private key SK, a keyword of interest $w' \in \mathcal{W}$ and outputs the trapdoor Q for the keyword w'.
- $\mathsf{Search}(UL, D, Q) \rightarrow$ search results or $\perp$: The search algorithm takes as input the user list UL, the index D and the user's trapdoor Q. It outputs valid search results or returns a search failure indicator $\perp$.

B. Construction for ABKS-UR

In this subsection, we will describe the concrete ABKS-UR construction from the viewpoint of system level based on the above defined algorithms. The system level operations include *System Setup*, *New User Enrollment*, *Secure Index Generation*, *Trapdoor Generation*, *Search*, and *User Revocation*. Notice that each individual system level operation may invoke one or more low level algorithms.

System Setup The TA calls the Setup algorithm to generate PK and MK. Specifically, it selects random elements $t_1, ..., t_{3n}$. Define a collision-resistant keyed hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, and its key is selected randomly and securely shared between owners and users (for simplicity, we use it without mentioning the secret key hereafter). Let $T_k = g^{t_k}$ for each $k \in \{1, ..., 3n\}$ such that for $1 \leq i \leq n$, T_i are referred to as *positive* attributes, T_{n+i} are for *negative* ones, and T_{2n+i} are thought of as *don't care*. Let Y be $e(g,g)^y$. The public key is $PK := \langle e, g, Y, T_1, ..., T_{3n} \rangle$ and the master key is $MK := \langle y, t_1, ..., t_{3n} \rangle$. The initial version number ver is 1. The TA publishes (ver, PK) with the signature of each component of PK, and retains (ver, MK).

New User Enrollment When receiving a registration request from a new legitimate user f, the TA first selects a random

$x_f \in \mathbb{Z}_p$ as a new MK component. Then, the TA generates a new PK component $Y'_f = Y^{x_f}$ and publishes it with its signature. After that, the KeyGen algorithm is called to create secret key SK for this user. For every $i \in \mathcal{N}$, the TA selects random r_i from $\mathbb{Z}_p$ hence $r = \sum_{i=1}^{n} r_i$. $\hat{K}$ is set as g^{y-r}. For $i \in \mathcal{S}$, set $K_i = g^{\frac{r_i}{t_i}}$ and $K_i = g^{\frac{r_i}{t_{n+i}}}$ otherwise. Finally, let F_i be $g^{\frac{r_i}{t_{2n+i}}}$. The secret key is $SK := \langle ver, x_f, \hat{K}, \{K_i, F_i\}_{i\in\mathcal{N}}\rangle$.

In addition, the server maintains a user list UL containing all the legitimate users' identity information for each dataset. Specifically, the data owner first selects a random element s from $\mathbb{Z}_p$. When a new user f joins in the system and is allowed to search the dataset, the data owner calls CreateUL algorithm to set $\bar{D}_f = Y_f'^{-s}$ and asks the CS to add the tuple $(ID_f, \bar{D}_f)$ into the user list, where ID_f is the identity of the user f.

Secure Index Generation Before outsourcing a file to the CS, the data owner calls EncIndex algorithm to generate a secure index D for this file. In particular, set $\hat{D} = g^s$ and $\tilde{D}$ to be Y^s. Given an access policy $GT = \bigwedge_{i\in\mathcal{I}} \underline{i}$, for each $i \in \mathcal{I}$, let $D_i = T_i^s$ if $\underline{i} = i$ and $D_i = T_{n+i}^s$ if $\underline{i} = \neg i$. For each $i \in \mathcal{N}\backslash\mathcal{I}$, let $D_i = T_{2n+i}^s$. For some attribute $i' \in \mathcal{N}$ (this fixed position can be seen as part of public parameter) and a keyword $w \in \mathcal{W}$, the data owner sets $D_{i'}$ to be $T_{i'}^{\frac{s}{H(w)}}$ with without loss of generality, attribute i' is assumed to be positive. The encrypted index $D := \langle ver, GT, \hat{D}, \tilde{D}, \{D_i\}_{i\in\mathcal{N}}\rangle$.

Trapdoor Generation Every legitimate user in the system is able to generate a trapdoor for any keyword of interest by calling the algorithm GenTrapdoor. Specifically, user f selects random $u \in \mathbb{Z}_p$. Let $\hat{Q} = \hat{K}^u$ and $\tilde{Q} = u + x_f$. Q_i is denoted as K_i^u and $Qf_i = F_i^u$. Thus, for the same i' in secure index generation phase, $Q_{i'}$ is set to be $K_{i'}^{H(w')\cdot u}$, where w' is the keyword of interest and $Qf_{i'} = F_{i'}^{H(w')\cdot u}$. The trapdoor $Q := \langle ver, \hat{Q}, \tilde{Q}, \{Q_i, Qf_i\}_{i\in\mathcal{N}}\rangle$, where ver is the version number of SK used for generating this trapdoor.

Search Upon receipt of a trapdoor Q and the user identity ID_f, 1) the CS finds out if ID_f exists on the user list of the target dataset. If not, the user is not allowed to search over the dataset; 2) otherwise, the CS continues the Search algorithm with the input of trapdoor Q, encrypted index D and $\bar{D}_f$ from the user list. We call this process *dataset search authorization*. Then, we move onto the fine-grained *file-level search authorization*, which includes three cases:

- If ver of Q is less than ver of D, it outputs $\perp$.
- If ver of Q is greater than ver of D, the algorithm ReEncIndex is called to update the index first.
- If ver of Q is equal to ver of D, the search process is performed as follows. For each attribute $i \in \mathcal{I}$, if $\underline{i} = i$ and $i \in S$, then

$$e(D_i, Q_i) = e(g^{t_i\cdot s}, g^{\frac{r_i\cdot u}{t_i}}) = e(g,g)^{s\cdot u\cdot r_i}.$$

If $\underline{i} = \neg i$ and $i \notin \mathcal{S}$, then

$$e(D_i, Q_i) = e(g^{t_{n+i}\cdot s}, g^{\frac{r_i\cdot u}{t_{n+i}}}) = e(g,g)^{s\cdot u\cdot r_i}.$$

For each $i \notin \mathcal{I}$,

$$e(D_i, Qf_i) = e(g^{t_{2n+i}\cdot s}, g^{\frac{r_i\cdot u}{t_{2n+i}}}) = e(g,g)^{s\cdot u\cdot r_i}.$$

For the attribute $i' \in \mathcal{N}$, $e(D_{i'}, Q_{i'})$ is equal to $e(g,g)^{s\cdot u\cdot r_{i'}}$ as well.

If the following equation holds, the user's attributes satisfy the access structure embedded in the index and $w' = w$,

$$\tilde{D}^{\tilde{Q}} \cdot \bar{D}_f \stackrel{?}{=} e(\hat{D}, \hat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*),$$

where $Q_i^* = Q_i$ if $i \in \mathcal{I}$ and $Q_i^* = Qf_i$ otherwise.

Correctness Provided that the user is authorized to access the file and $w' = w$, then

$$\begin{aligned} e(\hat{D}, \hat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*) &= e(g^s, g^{u\cdot y - u\cdot r}) \cdot \prod_{i=1}^{n} e(g,g)^{s\cdot u\cdot r_i} \\ &= e(g,g)^{s\cdot u\cdot y - s\cdot u\cdot r} \cdot e(g,g)^{s\cdot u\cdot r} \\ &= e(g,g)^{s\cdot u\cdot y} \\ &= Y^{s\cdot u} \\ &= Y^{s\cdot(x_f+u)} \cdot Y^{-s\cdot x_f} = \tilde{D}^{\tilde{Q}} \cdot \bar{D}_f. \end{aligned}$$

Discussion We can achieve scalable fine-grained file-level search authorization by data-owner-enforced attribute-based access structure on the index of each file. The search complexity is linear to the number of attributes in the system rather than the number of authorized users. Hence, this one-to-many authorization mechanism is more suitable for a large scale system, such as cloud. Moreover, the dataset search authorization by using a per-dataset user list may accelerate the search process, since the CS can decide whether it should go into a particular dataset or not. Otherwise, the CS has to search every file at rest.

User Revocation To revoke a user from current system, we re-encrypt the secure indexes stored on the server and update the remaining legitimate users' secret keys. Note that these tasks can be delegated to the CS using proxy re-encryption technique so that user revocation is very efficient. In particular, the TA adopts the ReKeyGen algorithm to generate the re-encryption key set $rk := \langle ver, \{rk_{i,val}\}_{i\in\mathcal{N},val\in\{+,-\}}\rangle$. Let attribute set Φ consist of the attributes that need to be updated, without which the leaving user's attributes will never satisfy the access policy. If an attribute $i \in \Phi$, $rk_{i,+} = \frac{t'_i}{t_i}$ is for the *positive* attribute i, and for the *negative* $rk_{i,-}$ is set to be $\frac{t'_{n+i}}{t_{n+i}}$, where both t'_i and t'_{n+i} are randomly selected from $\mathbb{Z}_p$. If $i \in \mathcal{N}\backslash\Phi$, set $rk_{i,val} = 1$, where $val \in \{+,-\}$. Then the TA refines the corresponding components in MK and PK, and publishes the new PK' with the signatures. The TA also sends rk and its signature to the CS.

After receiving rk from the TA, the server checks whether the version number ver in rk is equal to current ver of the system (or it can be greater than the current system ver in the case of lazy re-encryption, see *Discussion* below). If not, it discards this re-encryption key set. Otherwise, the CS verifies rk. Then, the server calls the ReEncIndex algorithm to re-encrypt the secure indexes in its storage with valid rk. Let Δ be the set including all the attributes in the access structures of secure indexes with the re-encryption keys not being 1 in rk.

For each positive $i \in \Delta$, D'_i is set as $D_i^{rk_{i,+}}$, or $D'_i = D_i^{rk_{i,-}}$ for negative ones. For $i \notin \Delta$, let D'_i be equal to D_i. Finally, the index is updated as $D' := \langle ver+1, GT, \hat{D}, \tilde{D}, \{D'_i\}_{i\in\mathcal{N}} \rangle$.

Furthermore, the server is able to update the remaining legitimate users' secret keys by the ReKey algorithm. Suppose that SKL is a list stored on the CS containing all the partial secret keys PSK's of all the legitimate users in the system. PSK is defined as $(ver, \{K_i\}_{i\in\mathcal{N}})$. Note that the CS cannot generate a valid trapdoor with PSK. Let Ω be the set including all the attributes in PSK with the re-encryption keys not being 1 in rk. For each attribute i in Ω, denote K'_i to be $K_i^{rk_{i,+}^{-1}}$ if i is positive and $K_i^{rk_{i,-}^{-1}}$ otherwise. For each $i \notin \Omega$, set $K'_i = K_i$. The updated $PSK' = (ver+1, \{K'_i\}_{i\in\mathcal{N}})$, which is returned to the legitimate user. User can also verify whether his/her secret key is the latest version by checking $e(T_i, K_i) = (T'_i, K'_i)$, where T'_i is the attribute component in the latest PK'. Here we suppose all the attributes i are positive. Otherwise, use T_{n+i} and T'_{n+i} instead in the equation.

Finally, the server may eliminate ID information of the revoked user f, i.e. the tuple $(ID_f, \bar{D}_f)$, from all the corresponding user lists.

Discussion To handle file index update efficiently, we could adopt the lazy re-encryption technique [15]. The CS stores the re-encryption key sets rk's and will not re-encrypt indexes until they are being accessed. Specifically, the CS could "aggregate" multiple rk's and deal with the index update in a batch manner. For instance, $ver = k$ in D, $ver = j$ in the latest rk and $k < j$, to re-encrypt the index, the CS just calls ReEncIndex once with $\prod_{\rho=k}^{j} rk_{i,val}^{(\rho)}$.

C. Conjunctive Keyword Search

Data user may prefer the returned files containing several intended keywords with one search request, which is referred to as conjunctive keyword search. Similar to [12], [13], our proposed ABKS-UR scheme is able to provide conjunctive keyword search functionality readily as follows. $D_{i'}$ is defined as $g^{\frac{s\cdot t_{i'}}{\prod_{w_j\in\mathcal{W}} H(w_j)}}$ or $g^{\frac{s\cdot t_{i'}}{\otimes_{w_j\in\mathcal{W}} H(w_j)}}$, where $\otimes$ denotes XOR operation. The components $Q_{i'}$ and $Qf_{i'}$ in the trapdoor are generated accordingly. It is worth noting that this method has almost the same efficiency as the single-keyword ABKS-UR scheme, regardless of the number of simultaneous keywords.

V. Security Analysis and Performance Evaluation

In this section, we analyze security properties of the proposed ABKS-UR scheme, and show that it achieves the defined security design goals. We then provide the performance evaluation on our proposed scheme.

A. Security Analysis

1) Keyword semantic security: In this paper, we formally define a semantic security game for ABKS-UR (see Appendix.A). We first give the following theorem, and then prove our ABKS-UR construction IND-sCP-CKA secure.

Theorem 1: If a probabilistic polynomial-time adversary wins the IND-sCP-CKA game with non-negligible advantage ϵ, then we can construct a simulator $\mathcal{B}$ to solve the DBDH problem with non-negligible advantage $\frac{\epsilon}{2}$.

Proof: See Appendix.B. ■

As per the above theorem, we can conclude that our proposed scheme is semantically secure in the selective model. Note that malicious users cannot launch collusion attack to generate a new valid secret key or trapdoor, which has been implicitly proved because the adversary $\mathcal{A}$ in our security game has the same capability as the malicious users, i.e. he can query different secret keys.

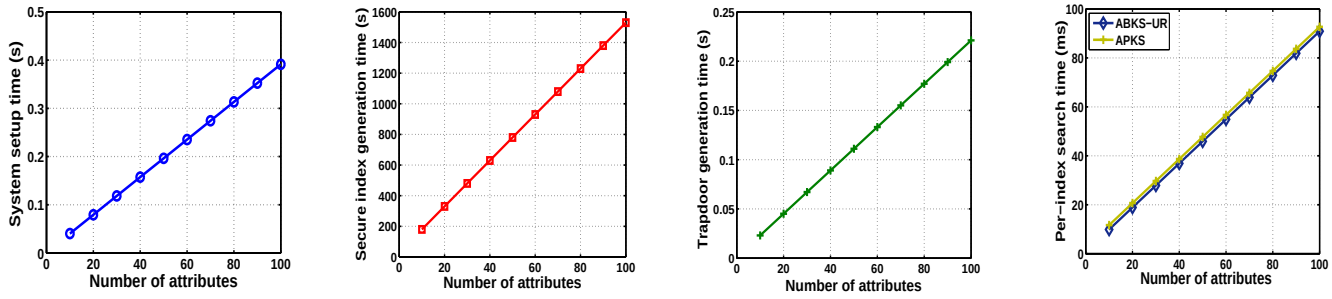
2) Trapdoor unlinkability: To generate a trapdoor, the user chooses a different random number u to obfuscate the trapdoor such that the CS is visually unable to differentiate two or more trapdoors even produced with the same keyword. Thus, the ABKS-UR can provide trapdoor unlinkability property.

B. Performance Evaluation

In this subsection, we will evaluate the performance of our proposed ABKS-UR scheme by real-world dataset and asymptotic computation complexity in terms of the pairing operation P, the group exponentiation E and the group multiplication M in $\mathbb{G}$, the group exponentiation E_1 and the group multiplication M_1 in $\mathbb{G}_1$. Note that we can realize the signature operation by any secure signature technique, e.g., RSA signature, which incurs fixed computation overhead, and here we only focus on evaluating the proposed ABKS-UR scheme, such that we do not consider the computation overhead for signature. We also ignore the hash operation as it is much more efficient than the above mentioned operations. Suppose there exist n attributes in the proposed scheme. The numerical performance evaluation is shown in Tab. I. Moreover, to evaluate the key operations of the proposed scheme, we use the real-world dataset, i.e. the Enron Email Dataset [27], which contains about half million files contributed from 150 users approximately. In the literature, there are few existing works on attribute-structure based authorized keyword search with experimental results. We will compare our ABKS-UR scheme with the predicate encryption based APKS scheme [20] in terms of search efficiency. We conduct our experiment using C and the Paring-Based Cryptography (PBC) Library [28] on a Linux Server with Intel Core i3 Processor 3.3GHz. We adopt the type A elliptic curve of 160-bit group order, which provides 1024-bit discrete log security equivalently.

1) System Setup: At this initial phase, the TA defines the public parameter, and generates PK and MK. The main computation overhead is $3n$ exponentiations in $\mathbb{G}$, one exponentiation in $\mathbb{G}_1$ and one pairing operation on the TA side. As shown in Fig.2 (a), the time cost for system setup is very efficient and is linear to the number of attributes in the system.

2) New User Enrollment: When a new legitimate user wants to join in the system, he/she has to request the TA to generate the secret key SK, which needs $2n+1$ exponentiations in $\mathbb{G}$. The TA also needs one exponentiation in $\mathbb{G}_1$ to generate a new PK component for the user. A data owner may also allow the user to access the dataset by adding him/her onto



(a) (b) (c) (d)

Fig. 2. Performance evaluation on ABKS-UR. (a) Time cost for system setup. (b) Secure index generation time for 10000 files. (c) Trapdoor generation time. (d) Time cost for search over a single index.

TABLE I
NUMERICAL EVALUATION OF ABKS-UR

Operation	Computation complexity
System Setup	$3n\mathsf{E} + \mathsf{E}_1 + \mathsf{P}$
New User Enrollment	$(2n+1)\mathsf{E} + 2\mathsf{E}_1$
Secure Index Generation	$(n+1)\mathsf{E} + \mathsf{E}_1$
Trapdoor Generation	$(2n+1)\mathsf{E}$
Per-index Search	$(n+1)\mathsf{P} + (n+2)\mathsf{M}_1 + \mathsf{E}_1$
ReKeyGen	$x(\mathsf{M}+\mathsf{E}), 1 \le x \le n$
ReEncIndex	$y\mathsf{E}, 1 \le y \le n$
ReKey	$z\mathsf{E}, 1 \le z \le n$

the corresponding user list, which incurs one exponentiation in $\mathbb{G}_1$. It is obvious that the time cost to enroll a new user is proportional to the number of attributes in the system.

3) Secure Index Generation: The data owner approximately needs $(n+1)\mathsf{E} + \mathsf{E}_1$ to generate a secure index for a file. Furthermore, we evaluate the practical efficiency of creating secure indexes for 10000 files, as shown in Fig.2 (b). It exhibits the expected linearity with the number of attributes in the system. When there exist 30 attributes in the system, the data owner would spend about 8 minutes encrypting the indexes for 10000 files. Note that this computational burden on the data owner is a one-time cost. After all the indexes outsourced to the CS, the following index re-encryption operation is also delegated to the server. Thus, the overall efficiency for encrypting index is totally acceptable in practice.

4) Trapdoor Generation: With the secret key, data user is free to produce the trapdoor of any keyword of interest, which requires about $2n+1$ group exponentiations in $\mathbb{G}$. Moreover, the experimental result in Fig.2 (c) shows that our proposed authorized keyword search scheme enjoys very efficient trapdoor generation. In accordance with the numerical computation complexity analysis, the trapdoor generation will need more time with the increased number of attributes.

5) Search: To search over a single encrypted index, the dominant computation of ABKS-UR is $n+1$ pairing operations, while APKS [20] needs $n+3$ pairing operations. Fig.2 (d) shows the practical search time of ABKS-UR and APKS on a single secure index with different number of attributes respectively. With the same number of system attributes, ABKS-UR is slightly faster than APKS. Moreover, compared with APKS, ABKS-UR allows users to generate trapdoors independently without resorting to an always online attribute authority, and it has a broader range of applications due to the arbitrarily-structured data search capability. Notice that the search complexity of our scheme will varies a lot for different data users, since the *dataset search authorization* only allows users on the user lists to further access the corresponding datasets. Assume that there exist 10000 files and 30 system attributes. In the worse case of search over every file in the storage, the CS, with the same hardware/softwore specifications as our experiment, requires less than 5 minutes to complete the search operation. Thus, with a more powerful cloud, our proposed ABKS-UR scheme would be efficient enough for practical use.

6) User Revocation: As the server can efficiently eliminate the revoked user's identity information from the corresponding user lists, we do not show it in Tab.I. Instead, we calculate the main computation complexity of ReKeyGen, ReEncIndex and ReKey. To update the system, the TA uses the algorithm ReKeyGen to produce the new version of MK' and PK', and the re-encryption key set rk. Depending on the number of attributes to be updated, generating rk requires minimum M to maximum $n\mathsf{M}$ operations. Likewise, the computation overhead for PK' is within the range from E to $n\mathsf{E}$. Moreover, the CS calls the ReEncIndex algorithm to re-encrypt the secure indexes at its storage. Each index update needs E to $n\mathsf{E}$ operations in $\mathbb{G}$, which is also the computation overhead range for the CS to update a legitimate user's secret key by algorithm ReKey.

VI. CONCLUSION

In this paper, we design the first attribute-based keyword search scheme in the cloud environment, which enables scalable and fine-grained owner-enforced encrypted data search supporting multiple data owners and data users. Compared with existing public key authorized keyword search scheme [13], our scheme could achieve system scalability and fine-grainedness at the same time. Different from search scheme [20] with predicate encryption, our scheme enables a flexible authorized keyword search over arbitrarily-structured

data. In addition, by using proxy re-encryption and lazy re-encryption techniques, the proposed scheme is better suited to the cloud outsourcing model and enjoys efficient user revocation. Moreover, we formally prove the proposed scheme semantically secure in the selective model.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC 61272457, the National Project 2012ZX03002003-002, the 863 Project 2012AA013102, the 111 Project B08038, the IRT1078, the FRF K50511010001, the NSFC 61170251, and the U.S. NSF grants CNS-1217889 and CNS-1338102.

REFERENCES

[1] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. of INFOCOM*. IEEE, 2010, pp. 1–9.
[2] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE TPDS*, vol. 24, no. 1, pp. 131–143, 2013.
[3] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 136–149.
[4] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of S&P*. IEEE, 2000, pp. 44–55.
[5] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits." in *USENIX Security Symposium*, vol. 201, no. 1, 2011.
[6] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.
[7] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of CCS*. ACM, 2006, pp. 79–88.
[8] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. of CCS*. ACM, 2012, pp. 965–976.
[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. of INFOCOM*. IEEE, 2011, pp. 829–837.
[10] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. of ASIACCS*. ACM, 2013, pp. 71–82.
[11] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Information Security Practice and Experience*. Springer, 2008, pp. 71–85.
[12] Y. Yang, H. Lu, and J. Weng, "Multi-user private keyword search for cloud computing," in *Proc. of CloudCom*. IEEE, 2011, pp. 264–271.
[13] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. of Pairing*. Springer, 2007, pp. 2–22.
[14] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. of EUROCRYPT*. Springer, 1998, pp. 127–144.
[15] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proc. of FAST*, vol. 42, 2003, pp. 29–42.
[16] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. of Eurocrypt*. Springer, 2004, pp. 506–522.
[17] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. of ACNS*. Springer, 2004, pp. 31–45.
[18] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of cryptography*. Springer, 2007, pp. 535–554.
[19] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. of EUROCRYPT*. Springer, 2008, pp. 146–162.
[20] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *Proc. of ICDCS*. IEEE, 2011, pp. 383–392.
[21] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of CCS*. ACM, 2006, pp. 89–98.
[22] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of S&P*. IEEE, 2007, pp. 321–334.
[23] L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *Proc. of CCS*. ACM, 2007, pp. 456–465.
[24] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. of ASIACCS*. ACM, 2010, pp. 261–270.
[25] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Theory of Cryptography*. Springer, 2009, pp. 457–473.
[26] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, 1998.
[27] W. W. Cohen, "Enron email dataset." [Online]. Available: https://www.cs.cmu.edu/ enron/
[28] "Pairing-based cryptography libray." [Online]. Available: http://crypto.stanford.edu/pbc/
[29] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. of CRYPTO*. Springer, 2001, pp. 213–229.

APPENDIX

A. Security Definition for ABKS-UR

We first give the cryptographic assumption that our scheme relies on.

Definition 2 (The DBDH Assumption [29]): Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. The DBDH assumption is that no probabilistic polynomial-time adversary $\mathcal{B}$ can distinguish the tuple $A = g^a, B = g^b, C = g^c, e(g,g)^{abc}$ from the tuple $A = g^a, B = g^b, C = g^c, e(g,g)^z$ with non-negligible advantage. The advantage of $\mathcal{B}$ is defined as follows,

$$|Pr[\mathcal{B}(A, B, C, e(g,g)^{abc}) = 0] - Pr[\mathcal{B}(A, B, C, e(g,g)^z) = 0]|$$

where the probability is taken over the random choice of the generator g, the random choice of a, b, c, z in $\mathbb{Z}_p$, and the random bits consumed by $\mathcal{B}$.

The semantic security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$ is defined as follows.

Init. The adversary $\mathcal{A}$ submits a challenge access policy GT, a version number ver^* and $ver^* - 1$ attribute sets $\{\Phi^{(\rho)}\}_{1 \le \rho \le ver^*-1}$ to the challenger $\mathcal{B}$.

Setup. The challenger $\mathcal{B}$ runs Setup$(\lambda, \mathcal{N})$ to obtain PK and MK for version 1. For each version $\rho \in \{1, ..., ver^* - 1\}$, $\mathcal{B}$ runs ReKeyGen(Φ, MK). Then he publishes $\{rk^{(\rho)}\}_{1 \le \rho \le ver^*-1}$ to $\mathcal{A}$, where $rk^{(\rho)}$ is defined as the re-encryption key set of version ρ. Given $\{rk^{(\rho)}\}_{1 \le \rho \le ver^*-1}$, the adversary $\mathcal{A}$ is able to compute PK for the corresponding version $\rho + 1$.

Phase 1. By submitting any keyword $w \in \mathcal{W}$, the adversary $\mathcal{A}$ is allowed to request the challenger $\mathcal{B}$ to generate trapdoors of any version from 1 to $ver*$ polynomial times (in λ). The only restriction is that the attribute set associated with each trapdoor query submitted by $\mathcal{A}$ does not satisfy the challenge access structure GT.

Challenge. Upon receipt of challenge keyword $w_0, w_1 \in \mathcal{W}$ of the same length from the adversary $\mathcal{A}$, $\mathcal{B}$ flips a random coin $\mu \in \{0, 1\}$ and get a challenge index $D_\mu \leftarrow EncIndex(PK, GT, w_\mu)$, where GT is the challenge access structure and PK is of version ver^*. $\mathcal{B}$ returns D_μ to $\mathcal{A}$.

Phase 2. Same as phase 1.

Guess. Adversary $\mathcal{A}$ submits his guess μ' of μ.

Definition 3 (IND-sCP-CKA Security): The proposed ABKS-UR scheme is IND-sCP-CKA secure if for all probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $Adv_{\mathcal{A}}^{IND-sCP-CKA}$ in winning the semantic security game is negligible.

$$Adv_{\mathcal{A}}^{IND-sCP-CKA} = Pr[\mu' = \mu] - \frac{1}{2}.$$

Notice that the trapdoor query oracle in Phase 1 implicitly includes the secret key query oracle, which may send the partial secret key (see section IV) back to the adversary. Since the adversary $\mathcal{A}$ is allowed to obtain all the re-encryption keys, he is able to update indexes, secret keys and trapdoors on his own such that we do not let challenger answer these queries in Phase 1 and Phase 2. Moreover, in the selective model, our semantic security game allows the adversary to query any keywords at Phase 1 and Phase 2 as long as the attribute sets associated with the queried trapdoors do not satisfy the challenge access policy GT.

B. Security Proof for ABKS-UR

In what follows, we will prove ABKS-UR construction IND-sCP-CKA secure.

Proof: The DBDH challenger first randomly chooses $a, b, c, z \in \mathbb{Z}_p$ and a fair coin $\nu \in \{0, 1\}$. It defines Z to be $e(g, g)^{abc}$ if $v = 0$, and $e(g, g)^z$ otherwise. Then the simulator $\mathcal{B}$ is given a tuple $(A, B, C, Z) = (g^a, g^b, g^c, Z)$ and asked to output ν. The simulator $\mathcal{B}$ now plays the role of challenger in the following game.

Init. In this phase, simulator $\mathcal{B}$ receives the challenge access structure $GT = \bigwedge_{i \in \mathcal{I}} \underline{i}$, a version number ver^* and $ver^* - 1$ attribute sets $\{\Phi^{(\rho)}\}_{1 \le \rho \le ver^*-1}$ from adversary $\mathcal{A}$.

Setup. For PK of version 1, Simulator $\mathcal{B}$ sets Y to be $e(A, B) = e(g, g)^{a \cdot b}$, which implicitly defines $y = a \cdot b$. Choose random $x = \theta \in \mathbb{Z}_p$ and define Y' to be $e(A, B)^\theta = e(g, g)^{a \cdot b \cdot \theta}$. For each $i \in \mathcal{N}$, $\mathcal{B}$ selects random $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p$, and outputs the following public parameters.

For $i \in \mathcal{I}$, $T_i = g^{\alpha_i}$, $T_{n+i} = B^{\beta_i}$, $T_{2n+i} = B^{\gamma_i}$ if $\underline{i} = i$; $T_i = B^{\alpha_i}$, $T_{n+i} = g^{\beta_i}$, $T_{2n+i} = B^{\gamma_i}$ if $\underline{i} = \neg i$.

For $i \notin \mathcal{I}$, $T_i = B^{\alpha_i}$, $T_{n+i} = B^{\beta_i}$, $T_{2n+i} = g^{\gamma_i}$.

For each attribute set $\Phi^{(\rho)}, 1 \le \rho \le ver^* - 1$, $\mathcal{B}$ generates the re-encryption key $rk^{(\rho)}$ and the PK of that version. For each attribute $i \in \Phi^{(\rho)}$, $rk_{i,val}^{(\rho)}$ where $val \in \{+, -\}$, is randomly selected from $\mathbb{Z}_p$. $T_i^{(\rho+1)} = (T_i^{(\rho)})^{rk_{i,+}^{(\rho)}}$, $T_{n+i}^{(\rho+1)} = T_{n+i}^{(\rho)}$, and $T_{2n+i}^{(\rho+1)} = T_{2n+i}^{(\rho)}$ if attribute i is positive. Otherwise, $T_i^{(\rho+1)} = T_i^{(\rho)}$, $T_{n+i}^{(\rho+1)} = (T_{n+i}^{(\rho)})^{rk_{i,-}^{(\rho)}}$, and $T_{2n+i}^{(\rho+1)} = T_{2n+i}^{(\rho)}$. Then, for each $i \notin \Phi^{(\rho)}$, set $rk_{i,val}^{(\rho)} = 1$ and the remaining public parameters of version $\rho + 1$ are the same with those of version ρ. Finally, simulator $\mathcal{B}$ publishes $rk^{(\rho)} = \langle \rho, \{rk_{i,val}^{(\rho)}\}_{i \in \Phi^{(\rho)}, val \in \{+,-\}} \rangle$ to $\mathcal{A}$.

Phase 1. Without loss of generality, assume that adversary $\mathcal{A}$ submits a keyword w_l and a set $\mathcal{S} \subseteq \mathcal{N}$ to $\mathcal{B}$ for version ρ, where $1 \le \rho \le ver^*$ and $\mathcal{S}$ does not satisfy GT. $\mathcal{B}$ uses the collision-resistant hash function to output $H(w_l) = h_l$. Since $\mathcal{S}$ does not satisfy GT, a witness attribute $j \in \mathcal{I}$ must exist. Thus, either $j \in \mathcal{S}$ and $\underline{j} = \neg j$, or $j \notin \mathcal{S}$ and $\underline{j} = j$. Without loss of generality, we assume $j \notin \mathcal{S}$ and $\underline{j} = j$.

Simulator $\mathcal{B}$ chooses random $\{r_i'\}_{1 \le i \le n} \in \mathbb{Z}_p$. Set $r_j = a \cdot b + r_j' \cdot b$ and $r_i = r_i' \cdot b$ if $i \neq j$. Denote $r = \sum_{i=1}^n r_i = a \cdot b + \sum_{i=1}^n r_i' \cdot b$. $\mathcal{B}$ defines u to be a random nubmer λ selected from $\mathbb{Z}_p$. As such, $\hat{Q}$ is defined to be $g^{y \cdot u - r \cdot u} = g^{-\sum_{i=1}^n r_i' \cdot b \cdot \lambda} = B^{-\sum_{i=1}^n r_i' \cdot \lambda}$. The $\tilde{Q}$ component of the trapdoor is defined to be $x + u = \theta + \lambda$.

By defining $rk_{i,val}^{(\rho)} = 1$ where $val \in \{+, -\}$ if $i \notin \Phi^{(\rho)}$, $\mathcal{B}$ could compute the followings for each $i \in \mathcal{N}$: for $2 \le \rho \le ver^*$, $T_i^{(\rho)} = (T_i^{(1)})^{rk_{i,+}^{(1)} \cdot rk_{i,+}^{(2)} \cdots rk_{i,+}^{(\rho-1)}} = (T_i^{(1)})^{\prod_{o=1}^{\rho-1} rk_{i,+}^{(o)}}$, and $T_{n+i}^{(\rho)} = (T_{n+i}^{(1)})^{rk_{i,-}^{(1)} \cdot rk_{i,-}^{(2)} \cdots rk_{i,-}^{(\rho-1)}} = (T_{n+i}^{(1)})^{\prod_{o=1}^{\rho-1} rk_{i,-}^{(o)}}$.

$\mathcal{B}$ denotes $R_i^{(\rho)} = \prod_{o=1}^{\rho-1} rk_{i,+}^{(o)}$ and $R_{n+i}^{(\rho)} = \prod_{o=1}^{\rho-1} rk_{i,-}^{(o)}$.

Simulator $\mathcal{B}$ sets $Q_j = A^{\frac{\lambda}{\beta_j \cdot R_{j+1}^{(\rho)}}} \cdot g^{\frac{r_j' \cdot \lambda}{\beta_j \cdot R_{j+1}^{(\rho)}}} = g^{\frac{a \cdot b + r_j' \cdot b}{b \cdot \beta_j \cdot R_{j+1}^{(\rho)}} \cdot \lambda} = g^{\frac{r_j \cdot u}{b \cdot \beta_j \cdot R_{j+1}^{(\rho)}}}$.

For $i \neq j$, 1) $i \in \mathcal{S}$. $Q_i = B^{\frac{r_i' \cdot \lambda}{\alpha_i \cdot R_i^{(\rho)}}} = g^{\frac{r_i \cdot u}{\alpha_i \cdot R_i^{(\rho)}}}$ if $i \in \mathcal{I} \wedge \underline{i} = i$; $Q_i = g^{\frac{r_i' \cdot \lambda}{\alpha_i \cdot R_i^{(\rho)}}} = g^{\frac{r_i \cdot u}{b \cdot \alpha_i \cdot R_i^{(\rho)}}}$ if $(i \in \mathcal{I} \wedge \underline{i} = \neg i) \vee i \notin \mathcal{I}$. 2) $i \notin \mathcal{S}$. $Q_i = B^{\frac{r_i' \cdot \lambda}{\beta_i \cdot R_{n+i}^{(\rho)}}} = g^{\frac{r_i \cdot u}{\beta_i \cdot R_{n+i}^{(\rho)}}}$ if $i \in \mathcal{I} \wedge \underline{i} = \neg i$; $Q_i = g^{\frac{r_i' \cdot \lambda}{\beta_i \cdot R_{n+i}^{(\rho)}}} = g^{\frac{r_i \cdot u}{b \cdot \beta_i \cdot R_{n+i}^{(\rho)}}}$ if $(i \in \mathcal{I} \wedge \underline{i} = i) \vee i \notin \mathcal{I}$.

Similarly, let $Qf_j = A^{\frac{\lambda}{\gamma_j}} \cdot g^{\frac{r_j' \cdot \lambda}{\gamma_j}} = g^{\frac{a \cdot b + r_j' \cdot b}{b \cdot \gamma_j} \cdot \lambda} = g^{\frac{r_j \cdot u}{b \cdot \gamma_j}}$. For $\{Qf_i\}_{i \neq j}$, we have 1) $i \in \mathcal{I}$. $Qf_i = g^{\frac{r_i' \cdot \lambda}{\gamma_i}} = g^{\frac{r_i \cdot u}{b \cdot \gamma_i}}$. 2) $i \notin \mathcal{I}$. $Qf_i = B^{\frac{r_i' \cdot \lambda}{\gamma_i}} = g^{\frac{r_i \cdot u}{\gamma_i}}$.

Without loss of generality, assume $i' \in \mathcal{S} \cap \mathcal{I}$ and $\underline{i'} = i'$. Simulator $\mathcal{B}$ sets $Q_{i'} = B^{\frac{r_{i'}' \cdot \lambda \cdot h_l}{\alpha_{i'} \cdot R_{i'}^{(\rho)}}} = g^{\frac{r_{i'} \cdot u \cdot H(w_l)}{\alpha_{i'} \cdot R_{i'}^{(\rho)}}}$.

Challenge. Upon receiving the challenge keywords w_0, w_1 from adversary $\mathcal{A}$, simulator $\mathcal{B}$ flips a random coin $\mu \in \{0, 1\}$ and then encrypts w_μ with the challenge gate GT. From the collision-resistant hash function H, simulator $\mathcal{B}$ obtains $H(w_\mu) = h_\mu$. For version ver^* and $i \in \mathcal{I}$, D_i is defined to be $C^{\alpha_i \cdot R_i^{(ver^*)}}$ if $\underline{i} = i$ and $C^{\beta_i \cdot R_{n+i}^{(ver^*)}}$ if $\underline{i} = \neg i$. For $i \notin \mathcal{I}$, let $D_i = C^{\gamma_i}$. Without loss of generality, assume $i' \in \mathcal{I}$ and $\underline{i'} = i'$ such that $D_{i'} = C^{\frac{\alpha_{i'} \cdot R_{i'}^{(ver^*)}}{h_\mu}}$. Finally, $\mathcal{B}$ sets $\hat{D} = C, \tilde{D} = Z$ and $\bar{D} = Z^{-\theta}$.

Phase 2. Same as phase 1.

Guess. Adversary $\mathcal{A}$ submits μ' of μ. If $\nu = 1$, adversary $\mathcal{A}$ cannot acquire any advantage in this semantic security game but a random guess. Therefore, we have $Pr[\mu \neq \mu' | \nu = 1] = \frac{1}{2}$. When $\mu \neq \mu'$, simulator $\mathcal{B}$ outputs $\nu' = 1$, such that $Pr[\nu' = \nu | \nu = 1] = Pr[\nu' = 1 | \nu = 1] = \frac{1}{2}$. If $\nu = 0$, a valid D is given to adversary $\mathcal{A}$. He can win this game with non-negligible advantage ϵ. Hence, $Pr[\mu = \mu' | \nu = 0] = \frac{1}{2} + \epsilon$. When $\mu = \mu'$, simulator $\mathcal{B}$ outputs $\nu' = 0$, we have $Pr[\nu' = \nu | \nu = 0] = Pr[\nu' = 0 | \nu = 0] = \frac{1}{2} + \epsilon$. The advantage $Adv_{\mathcal{A}}^{DBDH}$ of simulator $\mathcal{B}$ in the DBDH game is $Pr[\nu' = \nu] - \frac{1}{2} = Pr[\nu' = \nu | \nu = 1] Pr[\nu = 1] + Pr[\nu' = \nu | \nu = 0] Pr[\nu = 0] - \frac{1}{2} = \frac{1}{2} \cdot \frac{1}{2} + (\frac{1}{2} + \epsilon) \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}$ ■