7-2017

Attribute-based encryption with expressive and authorized keyword search

Hui CUI
Singapore Management University, hcui@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Joseph K. LIU
Monash University

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Citation

CUI, Hui; DENG, Robert H.; LIU, Joseph K.; and LI, Yingjiu. Attribute-based encryption with expressive and authorized keyword search. (2017). *Information Security and Privacy: ACISP 2017: 22nd Australasian Conference: Auckland, New Zealand, July 3-5: Proceedings*. 10342, 106-126.
Available at: https://ink.library.smu.edu.sg/sis_research/3816

Attribute-Based Encryption with Expressive and Authorized Keyword Search

Hui Cui[1(✉)], Robert H. Deng[1], Joseph K. Liu[2], and Yingjiu Li[1]

[1] School of Information Systems, Singapore Management University, Singapore, Singapore
{hcui,robertdeng,yli}@smu.edu.sg
[2] Faculty of Information Technology, Monash University, Melbourne, Australia
joseph.liu@monash.edu

Abstract. To protect data security and privacy in cloud storage systems, a common solution is to outsource data in encrypted forms so that the data will remain secure and private even if storage systems are compromised. The encrypted data, however, must be pliable to search and access control. In this paper, we introduce a notion of attribute-based encryption with expressive and authorized keyword search (ABE-EAKS) to support both expressive keyword search and fine-grained access control over encrypted data in the cloud. In ABE-EAKS, every data user is associated with a set of attributes and is issued a private attribute-key corresponding to his/her attribute set, and each data owner encrypts the message using attribute-based encryption and attaches the encrypted message with encrypted keywords related with the message, and then uploads the encrypted message and keywords to the cloud. To access encrypted messages containing certain keywords satisfying a search policy, a data user generates a trapdoor for the search policy using his/her private attribute-key and sends it to the cloud server equipped to the cloud. The cloud server searches over encrypted data stored in the cloud for the encrypted messages containing keywords satisfying the search policy and sends back the results to the data user who then decrypts the returned ciphertexts to obtain the underlying messages. We present a generic construction for ABE-EAKS, formally prove its security, give a concrete construction, and then extend the concrete ABE-EAKS scheme to support user revocation. Also, we implement the proposed ABE-EAKS scheme and its extension and study their performance through experiments.

Keywords: Cloud storage · Data security and privacy · Keyword search · Attribute-based encryption · Access control

1 Introduction

Consider a cloud storage system (e.g., [21,29,31]) that keeps personal health records (PHRs) provided by various medical institutions (i.e., data owners), in which all PHRs are stored in encrypted forms to protect data security and

privacy [30]. In order to facilitate data sharing, it is important for a cloud storage system to support powerful keyword search and scalable access control over the encrypted PHRs [46]. A straightforward approach meeting this requirement is to combine public key encryption mechanism and public-key based keyword search[1] such as public-key encryption with keyword search (PEKS) put forward by Boneh et al. [8], which allows a cloud server (equipped to the cloud) to search over encrypted PHRs on behalf of authorized data users (e.g., doctors, scientists) without learning any information about the underlying PHRs. Informally, in this combined approach, a PHR is encrypted using a public-key encryption scheme, the keywords associated with the PHR are encrypted using PEKS, and the ciphertext uploaded to the cloud is a concatenation of the "ciphertext" on the PHR and the "PEKS ciphertext" on the keywords associated with the PHR. To retrieve all encrypted PHRs containing certain keywords, a data user generates a "trapdoor" corresponding to the keywords and sends it to the cloud server such that the cloud server is able to spot and return all encrypted PHRs containing the specified keywords but learns nothing about the underlying PHRs.

However, a traditional public-key encryption scheme is a one-to-one encryption scheme targeted for decryption by a single data user, while encrypted messages in the cloud storage scenarios are expected to be accessed by groups of data users. Attribute-based encryption (ABE) [28,37,43,44] is widely believed as a promising solution for accomplishing fine-grained access control over encrypted data. In an ABE scheme, every data user is identified by a set of attributes and issued a private attribute-key associated with his/her attributes, every message is encrypted under an access structure, and any data user whose set of attribute satisfies the access structure ascribed to a ciphertext can decrypt this ciphertext.

Ideally, search policies should be expressive such that it can be expressed as conjunction, disjunction or any Boolean formulas. For example, in the aforementioned cloud storage system for PHRs, to find the relationship between "diabetes" and "age" or "weight", a researcher may submit a keyword search request with a search policy such as "(Illness: Diabetes AND (Age: 30 OR Weight: 100–200))"[2]. Unfortunately, most of the existing PEKS schemes only support single keyword search as in [8]. Though there are efforts in designing expressive keyword search (EKS) schemes that allows expressive keyword search policies (e.g., [10,13,24,32]), in all existing EKS schemes, data users need to send trapdoor generation requests on search policies to a trusted third party such as the key generation center (KGC), and then forward the trapdoors given by the KGC to the cloud server to conduct search over encrypted data. Relying on the KGC to generate trapdoors is not consistent with the standard PEKS notion in which trapdoors are generated by each data user himself/herself, and makes the KGC a bottleneck for both security and performance as it requires the KGC to be online all the time to answer requests of data users. There are authorized keyword

[1] In this paper, unless otherwise specified, all keyword search schemes we talk about are in the public-key setting.

[2] Note that in this paper, each keyword is divided into two parts N_i: W_i, where the former is the keyword name and the latter is the keyword value, e.g., Illness, Age, Weight are keyword names and Diabetes, 30, 100–200 are keyword values.

search (AKS) schemes (e.g., [20,39–41]) which authorize data users the capabilities of generating trapdoors by themselves, but existing solutions on AKS either lack the expressiveness in search polices or is inefficient due to the use of bilinear pairings over the composite-order groups. We note that most of the previous keyword search schemes are designed without taking message encryption into consideration, and yet it is known that simply combining a public-key encryption scheme for the message encryption and a keyword search scheme for the encryption of keywords may result in a solution subject to severe attacks [3].

Contributions. Motivated by the above observations, we propose a notion of attribute-based encryption with expressive and authorized keyword search (ABE-EAKS) to better meet the needs of cloud storage, which supports keyword search and access control over encrypted data in the setting of multiple data owners and multiple data users such as the cloud-based PHR system. Our goal is to design an ABE-EAKS scheme which simultaneously enables fine-grained access control and expressive keyword search over encrypted data without depending on a trusted third party to generate trapdoors. We compare our proposed ABE-EAKS scheme with existing constructions on AKS in Table 1.

Table 1. Comparison of properties among the AKS schemes.

	Expressiveness	Authorized keyword search	Bilinear group	Construction
AKS [39]	AND, OR gates	✓	Composite-Order	Concrete
AKS [40]	AND gates	✓	Prime-Order	Concrete
AKS [41]	AND gates	✓	Prime-Order	Concrete
AKS [20]	Single Keyword	✓	Prime-Order	Concrete
ABE-EAKS	AND, OR gates	✓	Composite-Order Prime-Order	Generic

We briefly summarize our contributions in this paper as follows.

- Firstly, we propose the notion of ABE-EAKS, which allows fine-grained access control and expressive keyword search over encrypted messages without relying on a trusted third party for the trapdoor generation.
- Secondly, we give a generic construction of ABE-EAKS which can be applied to transform ABE scheme and EKS scheme into a secure ABE-EAKS scheme, and formally prove its security. The main potential security vulnerability of an integrated ABE and EAKS scheme is the "swapping attack" [3] where an attacker can tamper with the ciphertext (which could be either the part on message encryption or the part on keyword encryption) stored in the cloud without being detected so that a privileged data user will not obtain the correct message. Thanks to the generic technique introduced by Fujisaki and

Okamoto [15] to achieve security in the integrated public-key and symmetric encryption schemes, we protect ABE-EAKS from swapping attacks by applying a similar approach as in [15] such that a data user can check whether a ciphertext has been modified when performing decryption operation on the ciphertext.
- Thirdly, we describe an instantiation of ABE-EAKS by applying concrete ABE and EKS schemes into the generic transformation, and extend the instantiation with an efficient user revocation mechanism which simultaneously improves decryption efficiency.
- Fourthly, we implement the instantiation and its extension to assess their performance.

1.1 Related Work

Attribute-Based Encryption. Sahai and Waters [37] first introduced attribute-based encryption (ABE). Later, Goyal et al. [17] formulated two complimentary forms of ABE: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In CP-ABE, a private attribute-key is associated with a set of attributes and a ciphertext is associated with an access structure, while the situation is reversed in KP-ABE. Nevertheless, we believe that KP-ABE is less flexible than CP-ABE because the access structure is determined once a data user's private attribute-key is issued[3]. Bethencourt, Sahai and Waters [7] proposed the first CP-ABE scheme, but it was secure under the generic group model. Cheung and Newport [11] presented a CP-ABE scheme secure under the standard model, but it only allowed the access structures in AND gates. A CP-ABE scheme with expressive access structures was put forth by Goyal et al. [16] based on the number theoretic assumption. Lewko et al. [25] put forward the first fully secure CP-ABE scheme, but it was in the composite-order groups. Rouselakis and Waters [36] gave a large universe CP-ABE scheme in the prime-order groups to improve the efficiency of ABE built from the composite-order groups while overcoming the limitation of bounded attribute space, but it was selectively secure.

Public-Key Encryption with Keyword Search. Since Boneh et al. [8] initiated the study of public-key encryption with keyword search (PEKS), many solutions [3,4,6,10,18,19,24,27,32,34,35,39,42,48–50] were proposed focusing on addressing three limitations in PEKS: (1) how to make PEKS secure against offline keyword dictionary guessing attacks; (2) how to support expressive search policies; and (3) how to achieve security in the integrated public-key encryption (PKE) and keyword search in the public-key setting. For the security against offline keyword dictionary guessing attacks, it requires that no adversary (including the cloud server) can learn keywords from a given trapdoor. To the best of our knowledge, such a security notion is very hard to be achieved in the public-key setting [38]. In terms of the expressive search policies, there are only a few expressive keyword search (EKS) schemes [10,14,24,32,39], but they either are

[3] In the rest of the paper, unless otherwise specified, what we talk about is CP-ABE.

expensive in implementations (e.g., [10,24,32,39]) or have limitations in security (e.g., [14]). Concerning the security of the integrated PKE scheme with keyword search scheme, there are solutions such as [3,50], but they only consider the security in the setting of the traditional public-key encryption schemes.

Authorized Keyword Search. Narayan, Gagné and Safavi-Naini [33] combined PEKS and ABE to create a secure electronic health record system, which provided both keyword search and access control mechanisms, but it failed to address the privacy issue of access control policies. Li et al. [27] put forth a notion of authorized private keyword search (APKS) in the setting of cloud storage and presented two concrete constructions on APKS, but their schemes were limited in applications since the search policies were defined and maintained by the trusted authorities. Sun et al. [40,41] proposed an attribute-based keyword search with fine-grained owner-enforced search authorization scheme, but it only supported access structures expressed in "AND" gates and search policies with conjunctive keywords. Shi et al. [39] presented a searchable encryption based on ABE to support fine-grained search and access control, but their scheme required each data user to ask a trusted trapdoor generation center to create trapdoors on search policies on behalf of himself/herself. Jiang et al. [20] introduced the notion of public-key encryption with authorized keyword search (PEAKS), but their construction of PEAKS could only be applied to single keyword search.

1.2 Organization

The remainder of this paper is organized as follows. In Sect. 2, we revisit the definitions to be used in this paper. In Sect. 3, after depicting the system architecture for ABE-EAKS, we present its security definition. In Sect. 4, we give a generic construction of ABE-EAKS, and prove its security. We conclude this paper in Sect. 5.

2 Preliminaries

In this section, we review some basic cryptographic notions and definitions that are to be used in this paper.

2.1 Bilinear Pairings

Let G be a group of a prime order p with a generator g. We define $\hat{e} : G \times G \rightarrow G_1$ to be a bilinear map if it has the following properties [9].

- Bilinear: for all $g \in G$, and $a, b \in Z_p$, we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.
- Non-degenerate: $\hat{e}(g, g) \neq 1$.

We say that G is a bilinear group if the group operation in G is efficiently computable and there exists a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \rightarrow G_1$ as above.

2.2 Access Structure and Linear Secret Sharing

Definition 1 ***Access Structures*** [26,45]**.** *Let* $\{P_1, ..., P_n\}$ *be a set of parties. A collection* $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}}$ *is monotone if* $\forall B, C$ *: if* $B \in \mathbb{A}$ *and* $B \subseteq C$*, then* $C \subseteq \mathbb{A}$*. An (monotone) access structure is a (monotone) collection* $\mathbb{A}$ *of non-empty subsets of* $\{P_1, ..., P_n\}$*, i.e.,* $\mathbb{A} \subseteq 2^{\{P_1,...,P_n\}} \setminus \{\emptyset\}$*. The sets in* $\mathbb{A}$ *are called the authorized sets, and the sets not in* $\mathbb{A}$ *are called the unauthorized sets.*

Definition 2 ***Linear Secret Sharing Schemes*** [26,45]**.** *Let* P *be a set of parties,* $\mathbb{M}$ *be a matrix of size* $l \times n$*, and* $\rho : \{1, ..., l\} \rightarrow P$ *be a function mapping a row to a party for labeling. A secret sharing scheme* Π *over a set of parties* P *is a linear secret-sharing scheme (LSSS) over* Z_p *if*

1. *The shares for each party form a vector over* Z_p*.*
2. *There exists a matrix* $\mathbb{M}$ *which has* l *rows and* n *columns called the share-generating matrix for* Π*. For* $i = 1, ..., l$*, the* x*-th row of matrix* $\mathbb{M}$ *is labeled by a party* $\rho(i)$*, where* $\rho : \{1, ..., l\} \rightarrow P$ *is a function that maps a row to a party for labeling. Considering that the column vector* $v = (\mu, r_2, ..., r_n)$*, where* $\mu \in Z_p$ *is the secret to be shared and* $r_2, ..., r_n \in Z_p$ *are randomly chosen, then* $\mathbb{M}v$ *is the vector of* l *shares of the secret* μ *according to* Π*. The share* $(\mathbb{M}v)_i$ *belongs to a party* $\rho(i)$*.*

It has been noted in [26] that every LSSS also enjoys the linear reconstruction property. Suppose that Π is an LSSS for access structure $\mathbb{A}$. Let $\mathbf{A}$ be an authorized set, and define $I \subseteq \{1, ..., l\}$ as $I = \{i | \rho(i) \in \mathbf{A}\}$. Then the vector $(1, 0, ..., 0)$ is in the span of rows of matrix $\mathbb{M}$ indexed by I, and there exist constants $\{w_i \in Z_p\}_{i \in I}$ such that, for any valid shares $\{v_i\}$ of a secret μ according to Π, $\sum_{i \in I} w_i v_i = \mu$. These constants $\{w_i\}$ can be found in polynomial time with respect to the size of the share-generating matrix $\mathbb{M}$ [5].

Boolean Formulas [26]**.** Access structures can also be described in terms of monotonic boolean formulas. LSSS access structures are more general, and can be derived from representations as boolean formulas. There are standard techniques to convert any monotonic boolean formula into a corresponding LSSS matrix. The boolean formula can be represented as an access tree, where the interior nodes are AND and OR gates, and the leaf nodes correspond to attributes. The number of rows in the corresponding LSSS matrix will be the same as the number of leaf nodes in the access tree.

2.3 Attribute-Based Encryption

An attribute-based encryption (ABE) scheme $\mathcal{ABE}$ [37] consists of a setup algorithm $\mathcal{ABE}$.Set(1^λ) which outputs the public parameter par and the master private key msk on input a security parameter λ, a key generation algorithm $\mathcal{ABE}$.KG(par, msk, $\mathbf{A}$) which outputs a private attribute-key $sk_{\mathbf{A}}$ on input the public parameter par, the master private key msk and an attribute set $\mathbf{A}$, an encryption algorithm $\mathcal{ABE}$.Enc(par, $\mathbb{A}$ m) which outputs a ciphertext CT on

input the public parameter par, an access structure $\mathbb{A}$ and a message m, and a decryption algorithm $\mathcal{ABE}.\mathrm{Dec}(par, sk_{\mathbf{A}}, \mathrm{CT})$ which outputs a message m or a failure symbol $\perp$ on input the public parameter par, a private key $sk_{\mathbf{A}}$ and a ciphertext CT.

An ABE scheme $\mathcal{ABE}$ is indistinguishable under chosen plaintext attacks (IND-CPA secure) if for any probabilistic polynomial time (PPT) adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage function

$$\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{ABE},\mathcal{A}}(\lambda) = \Pr\left[b' = b \left| \begin{array}{l} (par, msk) \leftarrow \mathcal{ABE}.\mathrm{Set}(1^\lambda);\ b \leftarrow \{0,1\} \\ (m_0, m_1, \mathbb{A}^*, st) \leftarrow \mathcal{A}_1^{\mathcal{ABE}.\mathrm{KG}(msk,\cdot)}(par) \\ \mathrm{CT}^* \leftarrow \mathcal{ABE}.\mathrm{Enc}(par, \mathbb{A}^*, m_b) \\ b' \leftarrow \mathcal{A}_2^{\mathcal{ABE}.\mathrm{KG}(msk,\cdot)}(par, m_0, m_1, \mathbb{A}^*, st, \mathrm{CT}^*) \end{array} \right. \right] - 1/2$$

is negligible in the security parameter λ, where $|m_0| = |m_1|$, st is the state information, and adversary $\mathcal{A}$ is not allowed to make key generation queries on attributes that can satisfy the challenge access structure $\mathbb{A}^*$.

2.4 Symmetric Encryption

A symmetric encryption (SE) scheme $\mathcal{SE}$ with a key space $\mathcal{K}$ is composed of an encryption algorithm $\mathcal{SE}.\mathrm{Enc}(K, m)$ which outputs a ciphertext CT on input a key K and a message m, and a decryption algorithm $\mathcal{SE}.\mathrm{Dec}(K, \mathrm{CT})$ which outputs m or a failure symbol $\perp$ on input a key K and a ciphertext CT [15].

Let st be the state information. A symmetric encryption scheme $\mathcal{SE}$ is secure under chosen plaintext attacks (IND-CPA secure), if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage function

$$\mathbf{Adv}^{\text{IND-CPA}}_{\mathcal{SE},\mathcal{A}}(\lambda) = \Pr\left[b' = b \left| \begin{array}{l} K \leftarrow \mathcal{K}; b \leftarrow \{0,1\} \\ (m_0, m_1, st) \leftarrow \mathcal{A}_1(1^\lambda) \\ \mathrm{CT}^* \leftarrow \mathcal{SE}.\mathrm{Enc}(K, m_b) \\ b' \leftarrow \mathcal{A}_2(par, m_0, m_1, st, \mathrm{CT}^*) \end{array} \right. \right] - 1/2$$

is negligible in the security parameter λ, where $|m_0| = |m_1|$.

2.5 Expressive Keyword Search

An expressive keyword search (EKS) scheme $\mathcal{EKS}$ [24] consists of a setup algorithm $\mathcal{EKS}.\mathrm{Set}(1^\lambda)$ which outputs the public parameter par and the master private key msk on input a security parameter λ, a trapdoor generation algorithm $\mathcal{EKS}.\mathrm{Trd}(par, msk, \mathbb{S})$ which outputs a trapdoor $T_{\mathbb{S}}$ on input the public parameter par, the master private key msk and a search policy $\mathbb{S}$, an encryption algorithm $\mathcal{EKS}.\mathrm{Enc}(par, \mathbf{W})$ which outputs a ciphertext CT on input the public parameter par and a set of keywords $\mathbf{W}$, and a test algorithm $\mathcal{EKS}.\mathrm{Tst}(par, T_{\mathbb{S}}, \mathrm{CT})$ which outputs 1 or 0 on input the public parameter par, a trapdoor $T_{\mathbb{S}}$ and a ciphertext CT.

Denote by st the state information. An expressive keyword search scheme $\mathcal{EKS}$ is indistinguishable under chosen keyword-set attacks (IND-CKA secure) if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage function

$$\mathbf{Adv}^{\text{IND-CKA}}_{\mathcal{EKS},\mathcal{A}}(\lambda) = \Pr\left[b' = b \;\middle|\; \begin{array}{l} (par, msk) \leftarrow \mathcal{EKS}.\text{Set}(1^\lambda);\ b \leftarrow \{0,1\} \\ (\mathbf{W}_0^*, \mathbf{W}_1^*, st) \leftarrow \mathcal{A}_1^{\mathcal{EKS}.\text{Trd}(msk,\cdot)}(par) \\ \text{CT}^* \leftarrow \mathcal{EKS}.\text{Enc}(par, \mathbf{W}_b^*) \\ b' \leftarrow \mathcal{A}_2^{\mathcal{EKS}.\text{Trd}(msk,\cdot)}(par, \mathbf{W}_0^*, \mathbf{W}_1^*, st, \text{CT}^*) \end{array} \right] - 1/2$$

is negligible in the security parameter λ, where $|\mathbf{W}_0^*| = |\mathbf{W}_1^*|$, and adversary $\mathcal{A}$ is not allowed to make trapdoor generation queries on keywords that can be satisfied by the challenge keyword set $\mathbf{W}_0^*$ or $\mathbf{W}_1^*$.

3 System Architecture and Security Model

In this section, we describe the framework and security definition of attribute-based encryption with expressive and authorized keyword search (ABE-EAKS).

3.1 System Architecture

The architecture of an ABE-EAKS scheme is shown in Fig. 1, which consists of data owners who outsource encrypted data and the associated keywords to the cloud, data users who are identified by different attributes and are privileged to access data in the cloud, a key generation center (KGC) who holds the master private key and publishes the public parameter and is responsible for generating private attribute-keys for data users in terms of their attributes, and a cloud for data storage which is equipped with a cloud server who executes search operations over encrypted data for data users. Suppose that a data owner Bob uploads to the cloud an encrypted document M along with m encrypted keywords N_1: W_1, ..., N_m: W_m (here N_i is the keyword name and W_i is the keyword value) using the public parameter, and an authorized data user Alice, who is issued with a private attribute-key generated by the KGC in terms of her attributes, wants to search for documents containing keywords that satisfy a search policy $\mathbb{S}$. In order to do so, Alice generates a trapdoor over the search policy $\mathbb{S}$ using her private attribute-key. Then, Alice forwards this trapdoor to the cloud server such that the cloud server is able to spot all ciphertexts that contain the keywords which satisfy the search policy $\mathbb{S}$ and can be decrypted by Alice. Finally, the cloud server sends the relevant ciphertexts back to Alice.

We assume that the KGC is a trusted entity. The cloud is pubic, and thus any ciphertexts stored in the cloud might be tampered with by any malicious party. The cloud server is assumed to be "honest-but-curious", i.e., it honestly follows the protocol but it is curious to learn the data stored in the cloud. Data owners are assumed to honestly encrypt their data as well as the associated keywords and upload the corresponding ciphertext to the cloud. Data users are not trusted, and

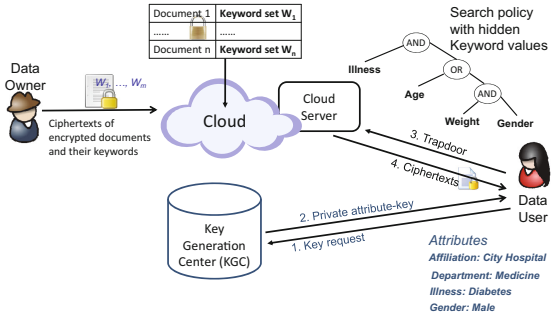


Fig. 1. System architecture of ABE-EAKS.

they may even collude with other participants in order to discover information beyond their privileges. We assume that the trusted KGC is equipped with a separate authentication mechanism to verify data users before issuing private attribute-keys to them.

3.2 Framework

Formally, an ABE-EAKS scheme consists of the following algorithms: setup algorithm Setup, user key generation algorithm KeyGen, trapdoor generation algorithm Trapdoor, encryption algorithm Encrypt, testing algorithm Test and decryption algorithm Decrypt. In an ABE-EAKS scheme, the KGC is given the public parameter and master private key generated from the Setup algorithm, and runs the KeyGen algorithm to generate each data user a private attribute-key in terms of his/her attributes. A data owner runs the Encrypt algorithm on the document and the relevant keywords using the public parameter, and uploads the corresponding ciphertext to the cloud. A data user can create a trapdoor on a search policy over a set of keywords by running the Trapdoor algorithm using his/her private attribute-key. Given a trapdoor, the cloud server runs the Test algorithm to determine whether an encrypted document contains the keywords satisfying the specified search policy and its access structure can be satisfied by the attributes associated with the trapdoor. After receiving the results from the cloud server, the data user runs the Decrypt algorithm on the ciphertexts to obtain the underlying document.

- $\text{Setup}(1^\lambda) \rightarrow (par, msk)$. Taking the security parameter λ as the input, this algorithm outputs the public parameter par and the master private key msk.
- $\text{KeyGen}(par, msk, \mathbf{A}) \rightarrow sk_{\mathbf{A}}$. Taking the public parameter par, the master private key msk and an attribute set $\mathbf{A}$ of a data user as the input, this algorithm outputs a private attribute-key $sk_{\mathbf{A}}$ for this data user.
- $\text{Trapdoor}(par, sk_{\mathbf{A}}, \mathbb{S}) \rightarrow T_{\mathbf{A},\mathbb{S}}$. Taking the public parameter par, the private attribute-key $sk_{\mathbf{A}}$ of a data user and a search policy $\mathbb{S}$ over a set of keywords as the input, this algorithm outputs a trapdoor $T_{\mathbf{A},\mathbb{S}}$.

- Encrypt(par, $(M, \mathbb{A})$, $\mathbf{W}$) $\rightarrow$ CT. Taking the public parameter par, a message M and an access structure $\mathbb{A}$, and a set of keywords $\mathbf{W}$ as the input, this algorithm outputs a ciphertext CT which consists of CT_M (an encryption of M under $\mathbb{A}$), $\mathrm{CT}_{\mathbf{W}}$ (an encryption of $\mathbf{W}$) and τ (a tag binding CT_M and $\mathrm{CT}_{\mathbf{W}}$ to prevent them from being tampered with).
- Test(par, CT, $T_{\mathbf{A},\mathbb{S}}$) $\rightarrow 1/0$. Taking the public parameter par, a ciphertext CT and a trapdoor $T_{\mathbf{A},\mathbb{S}}$ as the input, this algorithm outputs either 1 if the keywords associated with CT satisfies the search policy of $T_{\mathbf{A},\mathbb{S}}$ and the access structure ascribed to CT can be satisfied by the attributes of $T_{\mathbf{A},\mathbb{S}}$ or 0 otherwise, i.e., the Test algorithm outputs 1 if (1) the attributes associated with the trapdoor satisfy the access structure of the ciphertext; and (2) the ciphertext contains the keywords satisfying the search policy of the trapdoor.
- Decrypt(par, $sk_{\mathbf{A}}$, CT) $\rightarrow M/\perp$. Taking the public parameter par, a private attribute-key $sk_{\mathbf{A}}$ over an attribute set $\mathbf{A}$ and a ciphertext CT as the input. This algorithm parses $\mathrm{CT} = (\mathrm{CT}_M, \mathrm{CT}_{\mathbf{W}}, \tau)$, and checks whether the tag τ is valid for CT_M and $\mathrm{CT}_{\mathbf{W}}$. If so, it decrypts CT_M and outputs the plaintext M when the attributes of $sk_{\mathbf{A}}$ satisfies the access structure of CT_M. Otherwise, it outputs a failure symbol $\perp$.

We require that an ABE-EAKS scheme is correct, meaning that for all keyword sets $\mathbf{W}$ satisfying search policies $\mathbb{S}$, and attribute sets $\mathbf{A}$ satisfying access structures $\mathbb{A}$, if $(par, msk) \leftarrow \mathrm{Setup}(1^\lambda)$, $sk_{\mathbf{A}} \leftarrow \mathrm{KeyGen}(par, msk, \mathbf{A})$, $T_{\mathbf{A},\mathbb{S}} \leftarrow \mathrm{Trapdoor}(par, sk_{\mathbf{A}}, \mathbb{S})$, $\mathrm{CT} \leftarrow \mathrm{Encrypt}(par, (M, \mathbb{A}), \mathbf{W})$, then $\mathrm{Test}(par, \mathrm{CT}, T_{\mathbf{A},\mathbb{S}}) = 1$, $\mathrm{Decrypt}(par, sk_{\mathbf{A}}, \mathrm{CT}) = M$.

Notice that in the concrete construction, the input $\mathbb{A}$ in the Encrypt algorithm will be set to be $(\mathbb{M}_{\mathbb{A}}, \rho_{\mathbb{A}})$ where $\mathbb{M}_{\mathbb{A}}$ is a matrix, and $\rho_{\mathbb{A}}$ is a function maps the rows of $\mathbb{M}_{\mathbb{A}}$ to attributes. In addition, the input $\mathbb{S}$ in the Trapdoor algorithm will be set to be $(\mathbb{M}_{\mathbb{S}}, \rho_{\mathbb{S}}, \{\rho_{\mathbb{S}}(i)\})$, where $\mathbb{M}_{\mathbb{S}}$ is a matrix, and $\rho_{\mathbb{S}}$ is a function that associates the rows of $\mathbb{M}_{\mathbb{S}}$ to keyword names, and $\{\rho_{\mathbb{S}}(i)\}$ are the corresponding keyword values.

3.3 Security Definitions

In addition to provide the confidentiality of the encrypted data (i.e., data privacy), an ABE-EAKS scheme should ensure that any private information about the keywords will not be revealed from the ciphertext (i.e., keyword privacy). Also, it should guarantee that a ciphertext that encrypts a message and a set of keywords cannot be tampered with without being detected. Below we describe the security game called indistinguishability under chosen-ciphertext attacks (i.e., IND-CCA security) for ABE-EAKS to meet these requirements, which is defined between a challenger algorithm $\mathcal{C}$ and an adversary algorithm $\mathcal{A}$.

- Setup. Algorithm $\mathcal{C}$ runs the Setup algorithm to obtain the public parameter par and the master private key msk, and gives par to algorithm $\mathcal{A}$.
- Phase 1. Algorithm $\mathcal{A}$ adaptively issues the following queries.

1. Algorithm $\mathcal{A}$ issues queries for the private attribute-keys corresponding to the attribute sets $\mathbf{A}_1$, ..., $\mathbf{A}_{q_1}$. For each $\mathbf{A}_i$, $i \in [1, q_1]$, algorithm $\mathcal{C}$ runs the KeyGen algorithm to generate and send $sk_{\mathbf{A}_i}$ to algorithm $\mathcal{A}$.
2. Algorithm $\mathcal{A}$ issues queries for the plaintexts of the ciphertexts CT_1, ..., CT_{q_2}. For each CT_i, $i \in [1, q_2]$, algorithm $\mathcal{C}$ runs the Decrypt algorithm to output and send M_i to algorithm $\mathcal{A}$.

– Challenge. We describe this phase in terms of data privacy and keyword privacy, respectively.
  - Data privacy. Algorithm $\mathcal{A}$ outputs two messages M_0^*, M_1^* of the same size, an access structure $\mathbb{A}^*$ and a keyword set $\mathbf{W}^*$. Algorithm $\mathcal{C}$ randomly chooses $\beta \in \{0, 1\}$, runs the Encrypt algorithm on $(M_\beta^*, \mathbb{A}^*)$, $\mathbf{W}^*$ to obtain and send the challenge ciphertext CT^* to algorithm $\mathcal{A}$.
  - Keyword privacy. Algorithm $\mathcal{A}$ outputs a message M^*, an access structure $\mathbb{A}^*$ and two keyword sets $\mathbf{W}_0^*$, $\mathbf{W}_1^*$ of the same size. Algorithm $\mathcal{C}$ randomly chooses $\beta \in \{0, 1\}$, runs the Encrypt algorithm on $(M^*, \mathbb{A}^*)$, $\mathbf{W}_\beta^*$ to obtain and send the challenge ciphertext CT^* to algorithm $\mathcal{A}$.

– Phase 2. Algorithm $\mathcal{A}$ continues issuing queries to algorithm $\mathcal{C}$ as in Phase 1 except with the following restrictions.
1. Algorithm $\mathcal{A}$ issues queries for the private attribute-keys corresponding to the attribute sets $\mathbf{A}_{q_1+1}$, ..., $\mathbf{A}_q$ with the restriction that any $\mathbf{A}_i$ for $i \in [q_1 + 1, q]$ cannot satisfy $\mathbb{A}^*$.
2. Algorithm $\mathcal{A}$ issues queries for the plaintexts of the ciphertexts CT_{q_2+1}, ..., $\mathrm{CT}_{q'}$ with the restriction that any CT_i for $i \in [q_2 + 1, q']$ is not equal to CT^*.

– Guess. Algorithm $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$ and wins the game if $\beta' = \beta$.

An ABE-EAKS scheme is IND-CCA secure if the advantage function referring to the security game $\mathrm{Game}_{\Pi,\mathcal{A}}^{\mathrm{IND}}$

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathrm{IND}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[\beta = \beta'] - 1/2|$$

is negligible in the security parameter λ for any probabilistic polynomial-time (PPT) adversary algorithm $\mathcal{A}$.

In addition, an ABE-EAKS scheme is said to be selectively IND-CCA secure if an Init stage is added before the Setup phase where algorithm $\mathcal{A}$ commits to the challenge access structure $\mathbb{A}^*$ and keyword set $\mathbf{W}^*$ (or keyword sets $\mathbf{W}_0^*$, $\mathbf{W}_1^*$) which it aims to attack.

4 Generic Construction and Its Extensions

In this section, we give a generic construction of attribute-based encryption with expressive and authorized keyword search (ABE-EAKS), and analyze its security.

4.1 Generic Construction

Denote by $\mathcal{M}$ the message space, $\mathcal{K}$ the key space, $\mathcal{R}$ the randomness space. Let $\mathcal{ABE} = (\mathcal{ABE}.\text{Setup}, \mathcal{ABE}.\text{KeyGen}, \mathcal{ABE}.\text{Encrypt}, \mathcal{ABE}.\text{Decrypt})$ be an attribute-based encryption (ABE) scheme (e.g., [36,37]), $\mathcal{EKS} = (\mathcal{EKS}.\text{Setup}, \mathcal{EKS}.\text{Trapdoor}, \mathcal{EKS}.\text{Encrypt}, \mathcal{EKS}.\text{Test})$ be an expressive keyword search (EKS) scheme (e.g., [14,24]), and $\mathcal{SE} = (\mathcal{SE}.\text{Encrypt}, \mathcal{SE}.\text{Decrypt})$ be a symmetric encryption (SE) scheme. Below we describe the generic construction on ABE-EAKS.

- Setup. This algorithm takes the security parameter λ as the input. It runs the $\mathcal{ABE}$.Setup algorithm to obtain the public parameter $par_{\mathcal{ABE}}$ and the master private key $msk_{\mathcal{ABE}}$. Then, it runs the $\mathcal{EKS}$.Setup algorithm to obtain the public parameter $par_{\mathcal{EKS}}$ and the master private key $msk_{\mathcal{EKS}}$. Also, it randomly chooses two hash functions $H_0 : \mathcal{M} \to \mathcal{K}$, $H_1 : \mathcal{M} \to \mathcal{R}$. It outputs the public parameter $par = (par_{\mathcal{ABE}}, par_{\mathcal{EKS}}, H_0, H_1)$ and the master private key $msk = (msk_{\mathcal{ABE}}, msk_{\mathcal{EKS}})$.
  Remarks. For the correctness of the proposed generic construction on ABE-EAKS, we require that the schemes $\mathcal{EKS}$ and $\mathcal{ABE}$ share most elements in their public parameters such that $msk_{\mathcal{ABE}} \subseteq msk_{\mathcal{EKS}}$ (or $msk_{\mathcal{EKS}} \subseteq msk_{\mathcal{ABE}}$) holds. Note that this is possible since there exist techniques to convert a CP-ABE scheme to a KP-ABE scheme, and vice versa [2], and an EKS scheme can be obtained from a KP-ABE scheme [14,23]. Therefore, $ct_{\mathbf{W}}$ and $ct_{\mathbf{1}}$ (to be defined below) generated using the same randomness have several elements in common.
- KeyGen. This algorithm takes the public parameter par, the master private key msk and a data user's attribute set $\mathbf{A}$ as the input. It runs the $\mathcal{ABE}$.KeyGen algorithm on the attribute set $\mathbf{A}$ and outputs $sk_{\mathbf{A}}$ as the private attribute-key.
- Trapdoor. This algorithm takes the public parameter par, a private attribute-key $sk_{\mathbf{A}}$ and a search policy $\mathbb{S}$ as the input. It runs the $\mathcal{EKS}$.Trapdoor algorithm on the search policy $\mathbb{S}$ by using the private attribute-key $sk_{\mathbf{A}}$ in place of the master private key $msk_{\mathcal{EKS}}$ to generate the trapdoor $T_{\mathbf{A},\mathbb{S}} = (T_{\mathbf{A}}, T_{\mathbb{S}})$, where $T_{\mathbf{A}}$ is associated with attributes of the data user, and $T_{\mathbb{S}}$ is associated with the search policy.
  Remarks. Notice that there exists a twist here for running the $\mathcal{EKS}$.Trapdoor algorithm using the private attribute-key $sk_{\mathbf{A}}$ in place of the required master private key. Firstly, the Trapdoor algorithm randomly chooses a value s, and binds the value s to $sk_{\mathbf{A}}$ to obtain $T_{\mathbf{A}}$ by performing certain operations. Then it runs the $\mathcal{EKS}$.Trapdoor algorithm on the search policy $\mathbb{S}$ using the secret value s in place of the required master private key $msk_{\mathcal{EKS}}$ to obtain $T_{\mathbb{S}}$. Finally, it outputs the trapdoor $T_{\mathbf{A},\mathbb{S}} = (T_{\mathbf{A}}, T_{\mathbb{S}})$.
- Encrypt. This algorithm takes the public parameter par, a message M, an access structure $\mathbb{A}$ and a keyword set $\mathbf{W}$ as the input. Firstly, it randomly chooses $R \in \mathcal{M}$, and runs the $\mathcal{ABE}$.Encrypt algorithm on the "message" R and the access structure $\mathbb{A}$ to generate CT_R. Secondly, it computes CT_M by running the $\mathcal{SE}$.Encrypt algorithm on the message M using the key $H_0(R)$.

Thirdly, it computes $r = H_1(M, R)$, and runs the $\mathcal{EKS}$.Encrypt algorithm on the keyword set $\mathbf{W}$ using the randomness r to generate $ct_{\mathbf{W}}$. Fourthly, it runs the $\mathcal{ABE}$.Encrypt algorithm on an identity element $\mathbf{1}$ under the access structure $\mathbb{A}$ using the randomness r to generate $ct_{\mathbf{1}}$. Finally, it outputs the ciphertext $\mathrm{CT} = (\mathrm{CT}_R, \mathrm{CT}_M, \mathrm{CT}_{\mathbf{W}})$ for $\mathrm{CT}_{\mathbf{W}} = (ct_{\mathbf{W}}, ct_{\mathbf{1}})$, where $(\mathrm{CT}_R, \mathrm{CT}_M)$ is the encryption of the message M, $\mathrm{CT}_{\mathbf{W}}$ is the encryption of the keywords $\mathbf{W}$, which also implicitly plays the role of the tag τ.

- Test. This algorithm takes the public parameter par, a trapdoor $T_{\mathbf{A},\mathbb{S}}$ and a ciphertext CT as the input. It parses $T_{\mathbf{A},\mathbb{S}}$ as $(T_{\mathbf{A}}, T_{\mathbb{S}})$, and CT as $(\mathrm{CT}_R, \mathrm{CT}_M, (ct_{\mathbf{W}}, ct_{\mathbf{1}}))$. Firstly, it runs the $\mathcal{ABE}$.Decrypt algorithm on the ciphertext $ct_{\mathbf{1}}$ using $T_{\mathbf{A}}$ as the private attribute-key to obtain an intermediate value X_0. Then, it runs the $\mathcal{EKS}$.Test algorithm on the trapdoor $T_{\mathbb{S}}$ and the ciphertext $(X_0, ct_{\mathbf{W}})$. If the keywords and access structure ascribed to $\mathrm{CT}_{\mathbf{W}}$ satisfy the search policy and attributes associated with $T_{\mathbf{A},\mathbb{S}}$, it outputs 1. Otherwise, it outputs 0.
  Remarks. Since the attributes associated with the private attribute-key of each data user are embedded in the trapdoor, the Test algorithm also excludes those ciphertexts whose access structures cannot be satisfied by the attributes of the data user.
- Decrypt. This algorithm takes the public parameter par, a private attribute-key $sk_{\mathbf{A}}$ and a ciphertext CT as the input. It parses CT as $(\mathrm{CT}_R, \mathrm{CT}_M, \mathrm{CT}_{\mathbf{W}})$. It runs the $\mathcal{ABE}$.Decrypt algorithm on the ciphertext CT_R using the private attribute-key $sk_{\mathbf{A}}$ to obtains R'. Then, it computes M' by running the $\mathcal{SE}$.Decrypt algorithm on the ciphertext CT_M using the key $H_0(R')$. Finally, it computes $r' = H_1(M', R')$, and runs the Encrypt algorithm on using the randomness r' to obtain $\mathrm{CT}'_{\mathbf{W}}$. If $\mathrm{CT}'_{\mathbf{W}}$ is equal to $\mathrm{CT}_{\mathbf{W}}$, it outputs M. Otherwise, it outputs $\perp$.

4.2 Security Proof

Theorem 1. *Assuming that the underlying $\mathcal{ABE}$ is IND-CPA secure, $\mathcal{SE}$ is IND-CPA secure, and $\mathcal{EKS}$ is IND-CKA secure, then the proposed construction on ABE-EAKS is IND-CCA secure in the random oracle model.*

Proof. Assuming that there exists an adversary algorithm $\mathcal{A}$ that breaks the IND-CCA security of the proposed ABE-EAKS scheme, then we can build an adversary algorithm $\mathcal{A}'$ that breaks the IND-CPA security of the underlying schemes $\mathcal{ABE}$, $\mathcal{SE}$ or $\mathcal{EKS}$. Denote by $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$ the challenger algorithms in the IND-CPA security games of the schemes $\mathcal{ABE}$, $\mathcal{SE}$, $\mathcal{EKS}$, respectively.

- Setup. Algorithm $\mathcal{A}'$ is given $par_{\mathcal{ABE}}$ from algorithm $\mathcal{B}_0$ of $\mathcal{ABE}$, and $par_{\mathcal{EKS}}$ from the algorithm $\mathcal{B}_1$ of $\mathcal{EKS}$. Algorithm $\mathcal{A}'$ sends $par = (par_{\mathcal{ABE}}, par_{\mathcal{EKS}}, H_0, H_1)$ to algorithm $\mathcal{A}$, where H_0, H_1 are random oracles controlled by algorithm $\mathcal{A}'$.
- H_0, H_1-queries. At any time, algorithm $\mathcal{A}$ can query the random oracle H_0 (or H_1). To respond to these queries, algorithm $\mathcal{A}'$ keeps an initially empty list L_{H_0} (or L_{H_1}) of tuples (R_i, k_i) (or $((M_i, R_i), r_i)$).

- If the query R_i (or $(M_i,\ R_i)$) already exists in the list L_{H_0} (or L_{H_1}), algorithm $\mathcal{A}'$ responds with $k_i = H_0(R_i)$ (or $r_i = H_1(M_i, R_i)$).
- Otherwise, it randomly chooses k_i (or r_i), sets $k_i = H_0(R_i)$ (or $r_i = H_1(M_i, R_i)$), and stores k_i (or r_i) to the list L_{H_0} (or L_{H_1}).

– Phase 1. Algorithm $\mathcal{A}$ adaptively issue the following queries to algorithm $\mathcal{A}'$.
  - Algorithm $\mathcal{A}$ issues private attribute-key queries on attribute sets $\mathbf{A}_i$. Algorithm $\mathcal{A}'$ forwards each private attribute-key query on $\mathbf{A}_i$ to algorithm $\mathcal{B}_0$, and sends the corresponding private attribute-key obtained from algorithm $\mathcal{B}_0$ to algorithm $\mathcal{A}$.
  - Algorithm $\mathcal{A}$ issues decryption queries on ciphertexts CT_i. If algorithm $\mathcal{A}'$ does not have the private attribute-key to decrypt the ciphertext, it issues a private attribute-key query on an attribute set satisfying the access structure of CT_i to algorithm $\mathcal{B}_0$, and then uses the returned private attribute-key to decrypt CT_i and sends the result to algorithm $\mathcal{A}$. Otherwise, algorithm $\mathcal{A}'$ runs the $\mathcal{ABE}$.Decrypt algorithm on CT_i and sends the result to algorithm $\mathcal{A}$.

– Challenge. We discuss this phase in terms of data privacy and key privacy, respectively.
  - Data privacy. Algorithm $\mathcal{A}$ outputs two messages M_0^*, M_1^* of the same size, an access structure $\mathbb{A}^*$ and a keyword set $\mathbf{W}^*$. Algorithm $\mathcal{A}'$ sends R_0^*, R_1^* and $\mathbb{A}^*$ to algorithm $\mathcal{B}_0$ to obtain $\mathrm{CT}_{R_\beta^*}$, and M_0^*, M_1^* to algorithm $\mathcal{B}_1$ to obtain $\mathrm{CT}_{M_\beta^*}$. Also, it randomly chooses $k \in \mathcal{K}$, $r \in \mathcal{R}$, $\beta \in \{0,1\}$, sets $k = H_0(R_\beta^*)$, $r = H_1(M_\beta^*, R_\beta^*)$ (note that because of random oracle, adversary $\mathcal{A}'$ can easily perform this setting), and runs the $\mathcal{EKS}$.Encrypt algorithm on $\mathbf{W}^*$ to obtain $\mathrm{CT}_{\mathbf{W}^*}$ using the randomness r. Algorithm $\mathcal{A}'$ sends the challenge ciphertext $\mathrm{CT}^* = (\mathrm{CT}_{R_\beta^*},\ \mathrm{CT}_{M_\beta^*},\ \mathrm{CT}_{\mathbf{W}}^*)$ to algorithm $\mathcal{A}$.
  - Keyword privacy. Algorithm $\mathcal{A}$ outputs a message M^*, an access structure $\mathbb{A}^*$ and two keyword sets $\mathbf{W}_0^*$, $\mathbf{W}_1^*$ of the same size. Algorithm $\mathcal{A}'$ sends $\mathbf{W}_0^*$, $\mathbf{W}_1^*$ to algorithm $\mathcal{B}_2$ to obtain $\mathrm{CT}_{\mathbf{W}_\beta^*}$. Also, it randomly chooses $R^* \in \mathcal{M}$, $r \in \mathcal{R}$, sets $r = H_1(M^*, R^*)$ (assuming that r is the randomness used in generating $\mathrm{CT}_{\mathbf{W}_\beta^*}$), and runs the $\mathcal{ABE}$.Encrypt algorithm on R^* and $\mathbb{A}^*$ to obtain CT_{R^*}, the $\mathcal{SE}$.Encrypt algorithm on M^* using the key $H_0(R^*)$ to obtain CT_{M^*}. Algorithm $\mathcal{A}'$ sends the challenge ciphertext $\mathrm{CT}^* = (\mathrm{CT}_{R^*},\ \mathrm{CT}_{M^*},\ \mathrm{CT}_{\mathbf{W}_\beta}^*)$ to algorithm $\mathcal{A}$.

– Phase 2. Algorithm $\mathcal{A}$ continues issuing queries to algorithm $\mathcal{A}'$ as in Phase 1, following the restrictions defined in the security model.

– Guess. Algorithm $\mathcal{A}$ makes a guess β' for β, algorithm $\mathcal{A}'$ forwards β' to algorithm $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$ as the guesses to the IND-CPA security games for the schemes $\mathcal{ABE}$, $\mathcal{SE}$ and $\mathcal{KS}$.

In the view of algorithm $\mathcal{A}$, the simulation is the same as the real security game except that algorithm $\mathcal{A}$ issues R_β^* for $\beta \in \{0,1\}$ or R^* (or $(M_\beta^*,\ R_\beta^*)$ for $\beta \in \{0,1\}$ or $(M^*,\ R^*)$) to the random oracle H_0 (or H_1). Notice that algorithm $\mathcal{A}$ has negligible probability in outputting such queries; otherwise, it helps algorithm $\mathcal{A}'$ directly break the IND-CPA security of the underlying

attribute-based encryption scheme $\mathcal{ABE}$, symmetric encryption scheme $\mathcal{SE}$ or expressive keyword search scheme $\mathcal{EKS}$.

To conclude, if algorithm $\mathcal{A}$ can win the IND-CCA game of ABE-EAKS, then algorithm $\mathcal{A}'$ can win the IND-CPA game of the underlying schemes $\mathcal{ABE}$, $\mathcal{SE}$ or $\mathcal{EKS}$. This completes the proof of Theorem 1.

4.3 Extensions

The proposed generic construction on ABE-EAKS can be extended as follows.

- Standard model. Due to the efficiency purpose, the proposed ABE-EAKS scheme is secure in the random oracle model. There exist generic methodologies (e.g., [22]) to achieve security in the standard model, which can be applied to build a generic construction on ABE-EAKS that is secure in the standard model. Thus, we can replace the symmetric encryption scheme and random oracles in the proposed ABE-EAKS scheme by the tag-based encryption as introduced in [22], thereby resulting in a generic construction on ABE-EAKS that is secure in the standard model.
- User revocation. As a cloud storage system may involve a large number of data users whose access rights may change with time, it is important to equip it with an efficient user revocation mechanism. Taking efficiency into consideration, techniques utilizing a third party to achieve user revocation (e.g., [12,47]) might be desirable solutions, which can simultaneously reduce data users' computational overheads in decryption. It is possible to incorporate such techniques into ABE-EAKS to additionally achieve efficient user revocation in the cloud storage system. We detail how to accomplish it in full version[4] of this paper using a concrete construction as an example.

4.4 Performance Analysis

Let $l_{\mathbb{S}}$ be the number of keywords in a search policy, $l_{\mathbb{M}}$ be the number of attributes in an access structure, k be the size of an attribute set associated with a private attribute-key, and m be the size of a keyword set ascribed to a ciphertext. In Table 2, we summarize the computational overheads incurred in the instantiation of ABE-EAKS (given in the full version) and its extension supporting user revocation. Denote "NA" as not applicable, "E" as an exponentiation operation, "P" as a pairing operation, $I_{\mathbb{S}} = \{I_1, ..., I_{\chi_1}\}$ as a set of minimum keyword subsets satisfying a search policy $\mathbb{S}$, χ_2 as $|I_1| + ... + |I_{\chi_1}|$.

We implement the instantiation of ABE-EAKS and its extension in Charm [1]. We use Charm of version Charm-0.43 and Python 3.4 in our implementation. Along with Charm-0.43, we install the PBC library for the underlying cryptographic operations. Our experiments are run on a laptop with Intel Core i5-4210U CPU @ 1.70 GHz and 8.00 GB RAM running 64-bit Ubuntu 16.04. We conduct the experiments over the elliptic curves: SS512 and MNT159 to provide

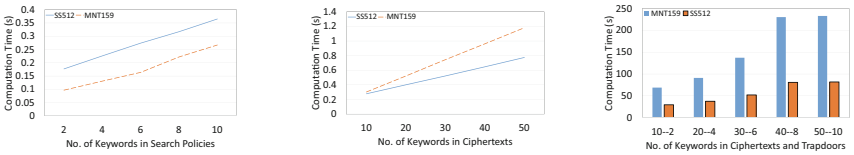
[4] Please contact the authors for the full version.

Table 2. Computational overhead of the instantiation and its extension.

	Trapdoor	Encrypt	Test (per search)	Transform server	Decrypt user
Instantiation on ABE-EAKS	$(4k+3)\cdot \mathrm{E} + 13l_{\mathbb{S}}\cdot \mathrm{E}$	$(4+10l_{\mathbb{M}})\cdot \mathrm{E} + 7m\cdot \mathrm{E}$	$\leq (k+\chi_2)\cdot \mathrm{E} + (3k+1+6\chi_2)\cdot \mathrm{P}$	NA	$\geq 4\cdot \mathrm{P} + 2\cdot \mathrm{E}$
Extension	$(4k+3)\cdot \mathrm{E} + 13l_{\mathbb{S}}\cdot \mathrm{E}$	$(4+10l_{\mathbb{M}})\cdot \mathrm{E} + 7m\cdot \mathrm{E}$	$\leq (k+\chi_2)\cdot \mathrm{E} + (3k+1+6\chi_2)\cdot \mathrm{P}$	$\geq 4\cdot \mathrm{P} + \mathrm{E}$	$2\cdot \mathrm{E}$

security level of 80-bit, where SS512 is a supersingular elliptic curve with the symmetric Type 1 pairing on it, and MNT159 is an asymmetric Type 3 pairing.

To begin with, we test the performance of the search function in the instantiation. In the experiments, each keyword contains a keyword name such as "Illness", "Position" and a keyword value such as "Diabetes", "Doctor", and we generate a random set of keywords containing 10 to 50 keywords, and use them to create 5,000 ciphertexts with access structures composed of 10 attributes. Thereafter, we create a set of search policies containing 2 to 10 keywords, and use them to yield trapdoors under the assumption that the data user is given a private attribute-key associated with 20 attributes. Finally, we run the test algorithm on the ciphertexts and the trapdoors.



(a) Average time for one trapdoor generation.

(b) Average time for one encryption operation.

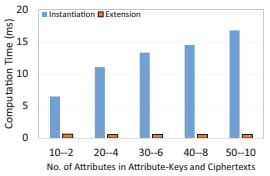
(c) Average time for testing among 5,000 ciphertexts.

Fig. 2. Computation time of the Trapdoor, Encrypt and Test algorithms.

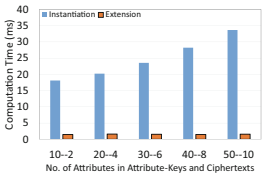
Figure 2 shows the average computation time of running the Trapdoor, Encrypt and Test algorithms, respectively. In terms of the trapdoor generation (See Fig. 2-(a)), the computation time of a data user in creating a trapdoor over a search policy of 2 to 10 keywords ranges from 0.2 s and 0.4 s for the SS512 curve, and 0.1 s to 0.3 s for the MNT159 curve, respectively. For the data encryption (See Fig. 2-(b)), the computation time of generating a ciphertext having 10 to 50 keywords and an access structure with 10 attributes is 0.3 s to 0.8 s for the SS512 curve, and 0.4 s to 1.2 s for the MNT159 curve, respectively. The computation time of the Test algorithm increases as the number of keywords involved in the trapdoor and the ciphertext raises (See Fig. 2-(c)). Regarding the 2 curves

used in our experiments, given a trapdoor for a search policy composed of 10 keywords, the computation time of searching over 5,000 encrypted documents each having 50 keywords is about 82 s and 233 s, respectively.

In addition, we test the computation time of a data user with 10 to 50 attributes in decrypting a ciphertext with an access structure composed of 2 to 10 attributes in the instantiation on ABE-EAKS and its extension in Fig. 3. In the instantiation of ABE-EAKS (See Fig. 3-(a)), the computation time of decrypting ciphertexts for access structures with 2 to 10 attributes using attribute-keys of 10 to 50 attributes ranges from 6.5 ms to 17 ms for the SS512 curve and 19 ms to 34 ms for the MNT159 curve, respectively, while in the extension (See Fig. 3-(b)), the computation time of decrypting ciphertexts for access structures with 2 to 10 attributes using attribute-keys of 10 to 50 attributes is about 0.6 ms in terms of the SS512 curve and 1.6 ms in terms of the MNT159 curve, respectively.



(a) Average time for the SS512 curve.



(b) Average time for the MNT159 curve.

Fig. 3. Computation time of decrypting a ciphertext by a data user.

5 Conclusions

Data encryption is an effective way for protecting data security and privacy in the cloud; however, in order for encrypted data to be useful, encryption mechanisms must be amenable to search and access control. In this paper, we introduced a notion of attribute-based encryption with expressive and authorized keyword search (ABE-EAKS) to support both expressive keyword search and fine-grained access control over encrypted data in cloud-based storage systems. We first presented the framework of ABE-EAKS and gave its security definition, and then provided a generic construction on ABE-EAKS which is able to transform any IND-CPA secure ABE scheme and IND-CKA secure EKS scheme into an IND-CCA secure ABE-EAKS scheme. Thereafter, we gave a concrete construction of ABE-EAKS based on the transformation and extended it to support user revocation. Finally, we implemented the concrete ABE-EAKS scheme and its extension, and studied their performance through experiments.

Acknowledgments. This research work is supported by the Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012 and the AXA Research Fund.

References

1. Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: A framework for rapidly prototyping cryptosystems. J. Cryptographic Eng. **3**(2), 111–128 (2013)
2. Attrapadung, N., Yamada, S.: Duality in ABE: Converting attribute based encryption for dual predicate and dual policy via computational encodings. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 87–105. Springer, Cham (2015). doi:10.1007/978-3-319-16715-2_5
3. Baek, J., Safavi-Naini, R., Susilo, W.: On the integration of public key data encryption and public key encryption with keyword search. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 217–232. Springer, Heidelberg (2006). doi:10.1007/11836810_16
4. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008. LNCS, vol. 5072, pp. 1249–1259. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69839-5_96
5. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Ph.D. thesis, Israel Institute of Technology, Israel Institute of Technology, June 1996
6. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74143-5_30
7. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (S& P 2007), pp. 321–334. IEEE Computer Society (2007)
8. Boneh, D., Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_30
9. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13
10. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70936-7_29
11. Cheung, L., Newport, C.C.: Provably secure ciphertext policy ABE. In: ACM Conference on Computer and Communications Security, CCS 2007, pp. 456–465. ACM (2007)
12. Cui, H., Deng, R.H., Li, Y., Qin, B.: Server-aided revocable attribute-based encryption. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 570–587. Springer, Cham (2016). doi:10.1007/978-3-319-45741-3_29
13. Cui, H., Deng, R.H., Wu, G., Lai, J.: An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures. In: Chen, L., Han, J. (eds.) ProvSec 2016. LNCS, vol. 10005, pp. 19–38. Springer, Cham (2016). doi:10.1007/978-3-319-47422-9_2
14. Cui, H., Wan, Z., Deng, R.H., Wang, G., Li, Y.: Efficient and expressive keyword search over encrypted data in cloud. IEEE Trans. Dependable Secure Comput. **PP**(99), 1 (2016)

15. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptol. **26**(1), 80–101 (2013)
16. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008). doi:10.1007/978-3-540-70583-3_47
17. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, CCS 2006, pp. 89–98 (2006)
18. Gu, C., Zhu, Y., Pan, H.: Efficient public key encryption with keyword search schemes from pairings. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 372–383. Springer, Heidelberg (2008). doi:10.1007/978-3-540-79499-8_29
19. Hwang, Y.H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73489-5_2
20. Jiang, P., Mu, Y., Guo, F., Wen, Q.: Public key encryption with authorized keyword search. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 2016. LNCS, vol. 9723, pp. 170–186. Springer, Cham (2016). doi:10.1007/978-3-319-40367-0_11
21. Jiang, T., Chen, X., Li, J., Wong, D.S., Ma, J., Liu, J.K.: Towards secure and reliable cloud storage against data re-outsourcing. Future Gener. Comput. Syst. **52**, 86–94 (2015)
22. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006). doi:10.1007/11681878_30
23. Lai, J., Deng, R.H., Li, Y.: Expressive CP-ABE with partially hidden access structures. In: ASIACCS 2012, pp. 18–19. ACM (2012)
24. Lai, J., Zhou, X., Deng, R.H., Li, Y., Chen, K.: Expressive search on encrypted data. In: ASIACCS 2013, pp. 243–252. ACM (2013)
25. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_4
26. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_31
27. Li, M., Yu, S., Cao, N., Lou, W.: Authorized private keyword search over encrypted data in cloud computing. In: ICDCS 2011, pp. 383–392. IEEE Computer Society (2011)
28. Liang, K., Au, M.H., Liu, J.K., Susilo, W., Wong, D.S., Yang, G., Yu, Y., Yang, A.: A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing. Future Gener. Comput. Syst. **52**, 95–108 (2015)
29. Liang, K., Susilo, W., Liu, J.K.: Privacy-preserving ciphertext multi-sharing control for big data storage. IEEE Trans. Inf. Forensics Secur. **10**(8), 1578–1589 (2015)
30. Liu, J., Huang, X., Liu, J.K.: Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption. Future Gener. Comput. Syst. **52**, 67–76 (2015)
31. Liu, J.K., Liang, K., Susilo, W., Liu, J., Xiang, Y.: Two-factor data security protection mechanism for cloud storage system. IEEE Trans. Comput. **65**(6), 1992–2004 (2016)

32. Lv, Z., Hong, C., Zhang, M., Feng, D.: Expressive and secure searchable encryption in the public key setting. In: Chow, S.S.M., Camenisch, J., Hui, L.C.K., Yiu, S.M. (eds.) ISC 2014. LNCS, vol. 8783, pp. 364–376. Springer, Cham (2014). doi:10.1007/978-3-319-13257-0_21
33. Narayan, S., Gagné, M., Safavi-Naini, R.: Privacy preserving EHR system using attribute-based infrastructure. In: ACM CCSW 2010, pp. 47–52. ACM (2010)
34. Rhee, H.S., Park, J.H., Lee, D.H.: Generic construction of designated tester public-key encryption with keyword search. Inf. Sci. **205**, 93–109 (2012)
35. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Improved searchable public key encryption with designated tester. In: ASIACCS 2009, pp. 376–379. ACM (2009)
36. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: ACM Conference on Computer and Communications Security, CCS 2013, pp. 463–474. ACM (2013)
37. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
38. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00457-5_27
39. Shi, J., Lai, J., Li, Y., Deng, R.H., Weng, J.: Authorized keyword search on encrypted data. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 419–435. Springer, Cham (2014). doi:10.1007/978-3-319-11203-9_24
40. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In: IEEE INFOCOM 2014, pp. 226–234. IEEE (2014)
41. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1187–1198 (2016)
42. Tang, Q., Chen, L.: Public-key encryption with registered keyword search. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 163–178. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16441-5_11
43. Wang, S., Liang, K., Liu, J.K., Chen, J., Yu, J., Xie, W.: Attribute-based data sharing scheme revisited in cloud computing. IEEE Trans. Inf. Forensics Secur. **11**(8), 1661–1673 (2016)
44. Wang, S., Zhou, J., Liu, J.K., Yu, J., Chen, J., Xie, W.: An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE Trans. Inf. Forensics Secur. **11**(6), 1265–1277 (2016)
45. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19379-8_4
46. Xhafa, F., Wang, J., Chen, X., Liu, J.K., Li, J., Krause, P.: An efficient PHR service system supporting fuzzy keyword search and fine-grained access control. Soft Comput. **18**(9), 1795–1802 (2014)
47. Yang, Y., Ding, X., Lu, H., Wan, Z., Zhou, J.: Achieving revocable fine-grained cryptographic access control over cloud data. In: Desmedt, Y. (ed.) ISC 2013. LNCS, vol. 7807, pp. 293–308. Springer, Cham (2015). doi:10.1007/978-3-319-27659-5_21
48. Yau, W., Phan, R.C., Heng, S., Goi, B.: Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. Int. J. Comput. Math. **90**(12), 2581–2587 (2013)

49. Zhang, B., Zhang, F.: An efficient public key encryption with conjunctive-subset keywords search. J. Netw. Comput. Appl. **34**(1), 262–267 (2011)
50. Zhang, R., Imai, H.: Generic combination of public key encryption with keyword search and public key encryption. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 159–174. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76969-9_11