KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage

Jiguo Li, Xiaonan Lin, Yichen Zhang and Jinguang Han, *Member, IEEE*

Abstract—Cloud computing becomes increasingly popular for data owners to outsource their data to public cloud servers while allowing intended data users to retrieve these data stored in cloud. This kind of computing model brings challenges to the security and privacy of data stored in cloud. Attribute-based encryption (ABE) technology has been used to design fine-grained access control system, which provides one good method to solve the security issues in cloud setting. However, the computation cost and ciphertext size in most ABE schemes grow with the complexity of the access policy. Outsourced ABE (OABE) with fine-grained access control system can largely reduce the computation cost for users who want to access encrypted data stored in cloud by outsourcing the heavy computation to cloud service provider (CSP). However, as the amount of encrypted files stored in cloud is becoming very huge, which will hinder efficient query processing. To deal with above problem, we present a new cryptographic primitive called attribute-based encryption scheme with outsourcing key-issuing and outsourcing decryption, which can implement keyword search function (KSF-OABE). The proposed KSF-OABE scheme is proved secure against chosen-plaintext attack (CPA). CSP performs partial decryption task delegated by data user without knowing anything about the plaintext. Moreover, the CSP can perform encrypted keyword search without knowing anything about the keywords embedded in trapdoor.

Index Terms—attribute-based encryption; cloud computing; outsourced key-issuing; outsourced decryption; keyword search

1 Introduction

CLOUD computing is a new computation model in which computing resources is regarded as service to provide computing operations. This kind of computing paradigm enables us to obtain and release computing resources rapidly. So we can access resource-rich, various, and convenient computing resources on demand [1].

The computing paradigm also brings some challenges to the security and privacy of data when a user outsources sensitive data to cloud servers. Many applications use complex access control mechanisms to protect encrypted sensitive information. Sahai and Waters [2] addressed this problem by introducing the concept for ABE. This kind of new public-key cryptographic primitive enables us to implement access control over encrypted files by utilizing access policies associated with ciphertexts or private keys. Two types of ABE schemes, namely key-policy ABE (KP-ABE) [3-8] and ciphertext-policy ABE (CP-ABE) [9-15] are proposed. For KP-ABE scheme, each ciphertext is related to a set of attributes, and each user's private key is associated with an access policy for attributes. A user is able to decrypt a ciphertext if and only if the attribute set related to the ciphertext satisfies the access policy associated with the user's private key. For CP-ABE scheme, the roles of an attribute set and an access policy are reversed. Bethencourt [9] et al. provided a CP-ABE scheme, which ensures encrypted data is kept confidential even if the storage server is untrusted. In order to withstand collusion attack and avoid sensitive information leakage from access structure, Qian et al. [11] proposed a privacy-preserving decentralized ABE scheme with fully hidden access structure. Deng et al. [12] constructed a ciphertext-policy hierarchical attribute-based encryption (CP-HABE) with short ciphertexts, which enables a CP-HABE system to host many users from different organizations by delegating keys. In CP-ABE scheme, a malicious user maybe shares his attributes with other users, which might leak his decryption privilege as a decryption blackbox due to financial profits. In order to solve above problem, Cao et al. [13-15] presented some traceable CP-ABE schemes, which can find the malicious users who intentionally leak the partial or modified decryption keys to others. One of the most efficiency drawbacks in the existing ABE schemes is time-consuming computation cost of key-issuing on TA side and decryption process on user side, which has turned into a bottleneck of the system. In order to solve the problem, some ABE schemes [16-27] have been proposed to outsource the expensive computation to CSPs, which greatly reduces the calculation overhead on user side. Since the data stored in CSPs becomes more and more numerous, traditional data utilization services will not work efficiently. An important issue is how to search useful information from very large data stored in CSPs. Boneh et al. [28] presented the notion of public key encryption with keyword search schemes which provides an approach to track over the encrypted data by keyword without leaking any information of keywords. However,

- *J. Li, X. Lin, and Y. Zhang are with the College of Computer and Information, Hohai University, Nanjing, China 211100. Email: lijiguo@hhu.edu.cn, 840448518@qq.com, zyc_718@163.com*
- *J. Han is with the Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, Jiangsu, China 210003. E-mail: jghan22@gmail.com.*

this scheme cannot support fine-grained access control on encrypted files. Some schemes [29-31] have been proposed to focus on the above problems. Qian et al. [32] provided a privacy preserving personal health record by utilizing multi-authority ABE.

1.1 Our Motivation and Contribution

It is well known that the shopping website has a lot of referral links which are collected by shopping website through the cookies. The cookies record the keywords that you often query. For example, if Alice likes to shop online and often browses the cosmetics and clothing, she often enters keywords like "cosmetics" and "clothing". Nevertheless, her interests will be exposed to the shop website since the cookies record the keywords of her interests. To solve the above issue, we generate the indexes for "cosmetics" and "clothing" in a secure manner. With the help of decryption cloud server provider and trapdoor associated with appointed keyword like "cosmetics", the user searches for the matching ciphertext without leaking the privacy of "cosmetics". In this way, we can protect the security and privacy of user’s interest through generating a trapdoor for each keyword in the form of encryption. D-CSP executes partial decryption task delegated by the user without knowing anything about the keyword, and the user retrieves the plaintext associated with the submitted keyword through local attribute private key.

We consider the case that the user Alice has a large number of data stored in the cloud. If Alice submits a request for accessing the encrypted data stored in the CSP, according to the traditional outsourced ABE scheme, the CSP downloads all the data, executes partial decryption and responses all corresponding data of Alice. This greatly increases the cost for communication and storage at Alice side. In this article, we organically integrate outsourced-ABE (OABE) with PEKS and present a novel cryptographic paradigm called outsourced attribute-based encryption scheme with keyword search function (KSF-OABE). In our system, when the user wants to outsource his sensitive information to the public cloud, he encrypts the sensitive data under an attribute set and builds indexes of keywords. As a result, the users can decrypt the ciphertext only if their access policies satisfy the corresponding attributes. By this way, when Alice submits the request with a trapdoor corresponding to a keyword “current”, CSP downloads all the data intended for Alice and just returns a partial ciphertext associated with the keyword “current”. Therefore, Alice can exclude the data what she does not hope to read.

1.2 Paper Organization

The paper is organized as follows. In Section 2, we review the preliminary knowledge including bilinear pairing, complexity assumption, secret sharing scheme and access structure which are used throughout the paper. In Section 3, we give our system model and security definition. ABE with keyword search and outsourcing decryption are presented in Section 4. The security of our scheme is proved in Section 5. We evaluate our construction in Section 6. Finally, we draw our conclusion in Section 7.

2 Preliminary Knowledge

We give some definitions and review related cryptographic knowledge about bilinear pairing, complexity assumption, access structures, and secret sharing scheme that our scheme relies on.

2.1 Notations

Table1 lists some notations utilized in this paper.

TABLE 1 Notations

Acronym	Description
TA	trusted authority
KG-CSP	key generation cloud server provider
D-CSP	decryption cloud server provider
S-CSP	storage cloud server provider
DO	data owner
DU	data user

2.2 Bilinear Pairing

Let G_1 and G_2 be multiplicative cyclic groups with prime order p. Suppose g is a generator of G_1. $e: G_1 \times G_1 \to G_2$ is a bilinear map if it satisfies the following properties:

(1) Bilinearity: For all $u, v \in G_1$, $e(u^a, v^b) = e(u,v)^{ab}$, where $a, b \in Z_p$ are selected randomly.

(2) Nondegeneracy: There exists $u, v \in G_1$ such that $e(u,v) \neq 1$.

(3) Computability: For all $u, v \in G_1$, there is an efficient algorithm to compute $e(u,v)$.

2.3 Complexity Assumption

Difinition1 (Decision Bilinear Diffie-Hellman Assumption). Let G_1 and G_2 be multiplicative cyclic groups with prime order p, and g be a generator of G_1. Given a tuple $(X,Y,Z) \in G_1$ where $X = g^x$, $Y = g^y$, $Z = g^z$, x, y, z are selected from Z_p randomly and T is selected from G_2 randomly. It’s difficult to decide whether $T = e(g,g)^{xyz}$.

2.4 Access Structures

Difinition2 (Access Structure) [16]. Suppose $\{P_1, ..., P_n\}$ are a set of parties. A collection $A \subseteq 2^{\{P_1,...P_n\}}$ is monotone if $\forall B, C$, $B \in A$ and $B \subseteq C$ then $C \in A$. A monotone access structure is a monotone collection A which is a non-empty subset for $\{P_1, ..., P_n\}$. The set in A is called an authorized set, and the set out of A is called an unauthorized set.

Let ω and A be an attribute set and access policy. A predicate $\gamma(\omega, A)$ is defined as follows: $\gamma(\omega, A) \in \{0,1\}$, if $\omega \in A$, the value of $\gamma(\omega, A)$ equals to 1, else the value is 0.

2.5 Secret Sharing Scheme

The secret sharing scheme [33] used in our paper is based on Lagrange interpolation method.

The Lagrange interpolation formula is as follows: $P_n(x) = \sum_{k=0}^{n} l_k(x) y_k = \sum_{k=0}^{n} (\prod_{j=0\, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}) y_k$, where the $l_k(x)$ is known as Lagrange coefficient and called basic function, the y_k is known as the interpolated function of kth insertion point used for sharing the secret. The $n-degree$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2542813, IEEE Transactions on Services Computing

polynomial $P_n(x)$ can be reconstructed through n+1 insertion points. The value of $P_n(0)$ will be kept as secret. In this paper, we select a random d-1 degree polynomial $P_{d-1}(x)$. The d insertion points are known as d attributes. We restore the $(d-1)-degree$ polynomial $P_{d-1}(x)$ through d shares y_i and the lagrange coefficient which is denoted as $\Delta_{i,S}=\prod_{j\in S, j\neq i}\frac{x-j}{i-j}$ where $|S|=d$. We obtain the secret $P_{d-1}(0)$ by computing $P_{d-1}(0)=\sum_{i\in S}\Delta_{i,S}(0)\cdot y_i$.

3 System Architecture, Formal Definition and Security Model

3.1 System Architecture

The system architecture for KSF-OABE scheme is shown as Figure 1, which involves the following participants.

Trusted Authority (TA). TA is the attribute authority center, which is responsible for the initialization of system parameters, and the generation of attribute private keys and trapdoor.

Key Generation Cloud Service Provider (KG-CSP). It is a participant that supplies outsourcing computing service for TA by completing the costly key generation tasks allocated by TA.

Decryption-Cloud Service Provider (D-CSP). It is a participant that supplies outsourcing computing service through accomplishing partial decryption for ciphertexts and keyword search service on the partially decrypted ciphertexts for data users who want to access the ciphertext.

Storage-Cloud Service Provider (S-CSP). It is a participant that supplies outsourcing data storage service for users who want to share file in cloud.

Data Owner (DO). This is a participant who intends to upload and share his data files on the cloud storage system in a secure way. The encrypted ciphertexts will be shared with intended receivers whose access structure will be satisfied by attribute set embedded in ciphertexts, that is to say the predicate $\gamma(\omega, A)=1$. The responsibility of DO is to generate indexes for some keywords and upload encrypted data with the indexes.

Data User (DU). This is a participant who decrypts the encrypted data stored in S-CSP with the help of D-CSP. If the attribute set for DU satisfies the access structures, DU is able to access the encrypted files and recover the original files from it. DU downloads intended ciphertexts with the help of trapdoor associated with appointed keyword. Data user is responsible for choosing keywords to create trapdoor, and decrypting data.

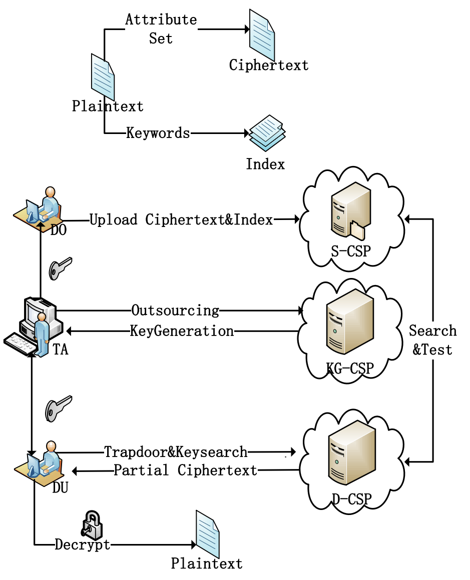

Fig. 1. System architecture

3.2 Formal Definition

We denote (I_{enc}, I_{key}) as the input of encryption and key generation. In our system, every user is bound up with access policy A, and every ciphertext is bound up with an attribute set ω. We have $(I_{enc}, I_{key})=(\omega, A)$ where ω and A are attribute set and access structure respectively. We describe the formal definition of KSF-OABE scheme as follow:

$Setup(\lambda)$: TA runs the $Setup$ algorithm, which takes a security parameter λ as input. It outputs the master secret key MSK and system public parameter PK. TA publishes the system public parameter PK and keeps the MSK secret. It is described as $Setup(1^\lambda)\rightarrow (PK, MSK)$.

$OABE-KeyGen_{init}(A, MSK)$: This algorithm is performed by TA which takes an access policy A and the master secret key MSK as input. It outputs a key pair (OK_{KGCSP}, OK_{TA}), where OK_{KGCSP} will be sent to KG-CSP to generate outsourcing private key and OK_{TA} will be kept by TA to compute local private key. It is described as $OABE-KeyGen_{init}(A, MSK)\rightarrow (OK_{KGCSP}, OK_{TA})$.

$OABE-KeyGen_{out}(A, OK_{KGCSP})$: This algorithm is performed by KG-CSP which takes an access policy A and OK_{KGCSP} as input. It outputs outsourcing private key SK_{KGCSP}. KG-CSP returns TA the SK_{KGCSP} . It is described as $OABE-KeyGen_{out}(A, OK_{KGCSP})\rightarrow SK_{KGCSP}$.

$OABE-KeyGen_{in}(OK_{TA})$: This algorithm is performed by TA which takes OK_{TA} as input. It outputs local private key SK_{TA}. It is described as $OABE-KeyGen_{in}(OK_{TA}) \rightarrow SK_{TA}$. TA then sets user private key as $SK=(SK_{KGCSP}, SK_{TA})$ and sends SK to user.

$KSF-KeyGen(PK, MSK, A, q_{BF})$: This algorithm is performed by TA which takes PK, MSK, an access policy A and q_{BF} as input. q_{BF} is a commitment value of blinding factor BF where the factor is generated by DU randomly. This algorithm outputs a query private key QK. It is described as $KSF-KeyGen(PK, MSK, A, q_{BF}) \rightarrow QK$.

$Encrypt(PK, M, \omega)$: This algorithm is run by DO which takes the system public parameter PK, a message M and an attribute set ω as input. It outputs a ciphertext CT. It is described as $Encrypt(PK, M, \omega) \rightarrow CT$.

$Index(PK, CT, KW)$: This algorithm is performed by DO which takes the system public parameter PK, and a keyword set $KW=\{kw_i\}_{i=1}^{m}$ as input. It outputs a searchable index of KW written as $IX(KW)$. It is described as $Index(PK, CT, KW) \rightarrow IX(KW)$.

$Trapdoor(PK, QK, BF, kw)$: This algorithm is performed by DU which takes the system public parameter PK, the query private key QK, the blinding factor BF and a keyword kw as input. It outputs a trapdoor T_{kw} corresponding to the keyword kw. It is described as $Trapdoor(PK, QK, BF, kw) \rightarrow T_{kw}$.

$Test(IX(KW), T_{kw}, CT)$: This algorithm is performed by D-CSP which takes the searchable indexes $IX(KW)$, a trapdoor T_{kw} bound up with an access policy A, and CT bound up with an attribute set ω as input. If the ω satisfies the access policy A embedded in CT. D-CSP partially decrypts the CT to get Q_{CT}. D-CSP searches for the corresponding ciphertexts CT related to the $IX(KW)$ through submitted trapdoor T_{kw}. It outputs a partial ciphertext Q_{CT}. The ciphertext CT which matches the keyword kw. It is described as $Test(IX(KW), T_{kw}, CT) \rightarrow Q_{CT}$.

$Decrypt(PK, CT, Q_{CT}, SK_{TA})$: It is run by DU which takes the system public parameter PK, the searched ciphertext CT, the partial decryption ciphertext Q_{CT}, and the local private key of DU as input. It outputs the plaintext M for the DU. It is described as $Decrypt(PK, CT, Q_{CT}, SK_{TA}) \rightarrow M$.

3.3 Security Model

Suppose that KG-CSP, S-CSP, and D-CSP are honest but curious. More accurately, they abide by the protocol, but try to obtain more information according to their ability. Moreover, curious users are permitted to collude with D-CSP and S-CSP. Two kinds of adversaries are described as follows:

$TypeI-Adversary$. This kind of adversary can be described as a curious user who colludes with D-CSP and S-CSP. The adversary is permitted to query the outsourcing private key SK_{KGCSP}, and the trapdoor T_{kw} of all users, and private key SK of dishonest users. The target for the adversary is to get any useful information on ciphertext and index of keywords which are not intended for him. The adversary should not get outsourcing key OK_{KGCSP} of any user.

$TypeII-Adversary$. This kind of adversary can be described as a curious KG-CSP. The adversary has the outsourcing keys OK_{KGCSP} of all users and tries to get some helpful information for the ciphertext stored in S-CSP. Note that, the KG-CSP searches for useful information from the ciphertext with OK_{KGCSP}, but it does not conclude with users in the proposed scheme.

The security requirement for our ABE scheme is similar to that proposed in [16]. We adopt a relaxation according to the secure notion called replayable CCA (RCCA) security in [34], which permits modifications to the ciphertext and they are not able to change the implied message in an effective way. We abide by RCCA security given above and define security for both $TypeI$ and $TypeII$ adversaries for KSF-OABE scheme. The RCCA security for our KSF-OABE is described as a game between a challenger and an adversary. The difference between our security model and that in [34] is that we define an additional game to simulate the $TypeII$ adversary with the outsourcing keys for all users. The game associated with $TypeI$ adversary is described as follow:

Setup: The challenger implements algorithm $Setup$ to obtain the public parameter PK and a master secret key MSK. It returns PK to the adversary $\mathcal{A}$ and keeps MSK secret.

Query Phase 1: The challenger initializes an empty table T. The adversary $\mathcal{A}$ repeatedly makes any of the following queries:

(1) $OABE-KeyGen_{out}$ query. On input an access policy A, the challenger searches the tuple $(A, SK_{KGCSP}, SK, T_{kw})$ in table T. If the tuple exists, it returns the outsourcing private key SK_{KGCSP} generated by KG-CSP. Otherwise, it runs the $OABE-KeyGen_{out}(A, OK_{KGCSP})$ algorithm to get SK_{KGCSP}. The challenger stores the outsourcing private key SK_{KGCSP} in table T and returns it to the adversary.

(2) $OABE-KeyGen_{in}$ query. According to an access policy A, the challenger searches the tuple $(A, SK_{KGCSP}, SK, T_{kw})$ in table T. If the tuple exists, it returns the private key SK. Else, it runs the algorithm $OABE-KeyGen_{out}(A, OK_{KGCSP})$ to get the SK_{KGCSP} and the $OABE-KeyGen_{in}(OK_{TA})$ algorithm to get local private key SK_{TA}. The challenger sets $SK=(SK_{KGCSP}, SK_{TA})$ and stores the private key SK in table T and returns it to the adversary.

(3) $Trapdoor$ query. According to an access policy A for trapdoor, the challenger searches the tuple $(A, SK_{KGCSP}, SK, T_{kw})$ in table T. If the tuple exists, it returns the trapdoor key T_{kw} associated with access policy A and a keyword used for search ciphertext. Otherwise, it runs the algorithm as above to get SK, runs the $KSF-KeyGen(PK, MSK, A, q_u)$ algorithm for QK and runs the $Trapdoor(PK, QK, BF, kw)$ algorithm to get the T_{kw}. The challenger stores the trapdoor in table T and returns the trapdoor key T_{kw} to adversary $\mathcal{A}$.

(4) *Decrypt* query. On input an access policy A and ciphertext (CT,Q_{CT}), the challenger queries the tuple (A,SK_{KGCSP},SK,T_{kw}) in table T. If the tuple exists, it implements $Decrypt(PK,CT,Q_{CT},SK)$ and returns M to the adversary. Else, it returns $\perp$.

Challenge: The adversary sends two messages M_0, M_1 with equal-length and a challenge attribute set ω^* to the challenger, subject to the restriction that, ω^* can not satisfy A A. The challenger chooses $\mu \in \{0,1\}$, and runs $Encrypt(PK,M_\mu,\omega^*) \to CT^*$. The challenger returns the challenge ciphertext CT^* to the adversary.

Query Phase 2. The adversary continues to adaptively query $OABE-KeyGen_{out}$, $OABE-KeyGen_{in}$, $Trapdoor$, and $Decrypt$ queries as in Query Phase 1 with the restrictions as follows:

(1) The adversary should not launch the $OABE-KeyGen_{out}$ and $OABE-KeyGen_{in}$ query that would result in an access structure A which will be satisfied by attributes set ω^*.

(2) The adversary should not issue the *Decrypt* query that the result will be M_0 or M_1.

Guess. The adversary gives a guess $\mu' \in \{0,1\}$ for μ.

The advantage which the adversary can win the game is defined as $|Pr(\mu'=\mu)-\frac{1}{2}|$.

The game associated with *TypeII* adversary is similar to the game described above.

Definition 3: An OABE scheme is RCCA-secure if any polynomial time adversary has at most negligible advantage winning in this security game, namely $|\Pr(u'=u)-\frac{1}{2}|<\varepsilon$.

CPA Security: A CP-ABE scheme that supports outsourcing key-issuing, decryption and keyword search function is CPA-secure if the adversary cannot launch *Decryption* queries in above game.

Selective Security: A CP-ABE scheme that supports outsourcing key-issuing, decryption and keyword search function is selectively secure if the adversary must submit the challenger attribute set ω^* prior to seeing the public parameters.

4 KSF-OABE Scheme

Our scheme is based on the OABE proposed in [16]. We use tree-based access structure described as in [16]. A is an tree-based access policy bound up with user private key, ω is an attribute set embedded in ciphertext, U is the attribute universe, and d is a threshold value set in advance. If $\gamma(\omega,A)=1$, S is a attribute set which satisfies $S \subseteq \{\omega \bigcap A\} \wedge |S|=d$. Based on the structure of the above, we add a function called keyword search function [28].

$Setup(\lambda)$: TA chooses multiplicative cyclic groups G_1,G_2 with prime order p, g is a generator of G_1. TA selects a bilinear map $e:G_1\times G_1\to G_2$ and defines the attributes in U as values in Z_p. For simplicity, we set $n=|U|$ and take the first n values in Z_p to be the attribute universe. TA randomly selects an integer $x\in Z_p$, computes $g_1=g^x$, and chooses $g_2,h,h_1,...,h_n\in G_1$ randomly where n is the number of attributes in universe. $H_1:\{0,1\}^*\to G_1$ and $H_2:G_2\to\{0,1\}^{\log p}$ are two secure hash functions. TA publishes $PK=(G_1,G_2,g,g_1,g_2,h,h_1,...,h_n,H_1,H_2)$ as system public parameter, and keeps the master secret key $MSK=x$ secret.

$OABE-KeyGen_{init}(A,MSK)$: Upon receiving a private key request on access policy A, TA selects $x_1\in Z_p$ randomly and computes $x_2=x-x_1 \bmod p$. $OK_{KGCSP}=x_1$ is sent to KG-CSP to generate outsourcing private key SK_{KGCSP}. $OK_{TA}=x_2$ is used to generate local private key SK_{TA} at TA side.

$OABE-KeyGen_{out}(A,OK_{KGCSP})$: TA sends OK_{KGCSP} to KG-CSP for generating outsourcing private key SK_{KGCSP}. Upon receiving the request on (A,OK_{KGCSP}), KG-CSP chooses a $(d-1)$-degree polynomial $q(\cdot)$ randomly such that $q(0)=x_1$. For $i\in A$, KG-CSP chooses $r_i\in Z_p$ randomly, and computes $d_{i0}=g_2^{q(i)}(g_1h_i)^{r_i}$ and $d_{i1}=g^{r_i}$. KG-CSP sends outsourcing private key $SK_{KGCSP}=\{d_{i0},d_{i1}\}_{i\in\omega}$ to TA.

$OABE-KeyGen_{in}(OK_{TA})$: TA takes OK_{TA} as input and computes $d_{\theta 0}=g_2^{x_2}(g_1h)^{r_\theta}$ and $d_{\theta 1}=g^{r_\theta}$, where $r_\theta\in Z_p$ is selected randomly, θ is the default attribute. TA sets private key $SK=(SK_{KGCSP},SK_{TA})$, where $SK_{TA}=\{d_{\theta 0},d_{\theta 1}\}$. TA responses the user with SK by secure channel.

$KSF-KeyGen(PK,MSK,A,q_{BF})$: To get a query private key of DU with access policy A, DU and TA interacts as follow:

—DU chooses a blinding factor $BF=u\in Z_p^*$ randomly, and provides a commitment $q_{BF}=g_2^{1/u}$ and an access policy A to TA. DU keeps u secret.

—TA retrieves $(g_1h)^{r_\theta}$ corresponding to A, and computes a query private key $QK=g_2^{x/u}(g_1h)^{r_\theta}$ for the DU.

—TA sends the query private QK to DU by secure channel.

$Encrypt(M,PK,\omega)$: It takes as input a message $M\in G_2$, the public parameters PK and an attribute set ω associated with ciphertext. DO randomly selects $s\in Z_p$ and calculates $C_0=Me(g_1,g_2)^s$, $C_1=g^s$, $C_i=(g_1h_i)^s$ for each $i\in\omega$, $C_\theta=(g_1h)^s$. DO outputs the ciphertext with attribute set ω, where $CT=(\omega\bigcup\{\theta\},C_0,C_1,\{C_i\}_{i\in\omega},$ $C_\theta)$.

$Index(PK,CT,KW)$: DO selects $r\in Z_p$ randomly and runs the index generation algorithm to compute $k_i=e(g_1,g_2)^s\cdot e(g,H_1(kw_i))^s\in G_2$ for each $kw_i\in KW$ where

$i=1,...,m$. DO outputs the indexes of keywords set as $IX(KW)=(K_1,K_2,K_i)$ for $kw_i \in KW$ where $K_1=C_1=g^s$, $K_2=C_\theta=(g_1h)^s$, $K_i=H_2(k_i)$. DO uploads the tuple $(CT, IX(KW))$ to the S-CSP.

$Trapdoor(PK,QK,BF,kw)$: In order to generate a trapdoor for a keyword kw, DU computes $T_q(kw)=H_1(kw)QK^u$, and sets $I=(I_{i0}=d_{i0}, I_{i1}=d_{i1})$ for all $i \in A$, $D_1={d_{\theta 1}}^u$. DU sets trapdoor for the keyword kw as $T_{kw}=(T_q(kw), I, D_1)$.

$Test(IX(KW),T_{kw},CT)$: DU submits a keyword search request by sending a trapdoor T_{kw} for keyword kw along with an access policy A which is bound up with private key for DU. If the attribute set embedded ciphertext satisfies the access policy A, D-CSP downloads all those ciphertexts and executes partial decryption for them. D-CSP computes:

$$Q_{CT}=\frac{\prod_{i\in S} e(C_1,I_{i0})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(I_{i1},C_i)^{\Delta_{i,S}(0)}}=e(g,g_2)^{sx_1}.$$

D-CSP searches for the corresponding ciphertext CT related to the appointed index of keywords through submitted trapdoor T_{kw}. D-CSP computes:

$$k_{kw}=\frac{e(K_1,T_q(kw))}{e(D_1,K_2)}=e(g_1,g_2)^s\cdot e(g,H_1(kw))^s,$$

and $H_2(k_{kw})$. D-CSP obtains the matching ciphertext by comparing $H_2(k_{kw})$ with each tuple $(CT, IX(KW))$ stored in S-CSP. D-CSP tests whether $H_2(k_i)=H_2(k_{kw})$ for each $kw_i \in KW$. D-CSP outputs $\perp$ if does not find matched tuple, otherwise D-CSP sends the search result that includes the tuple $(CT, IX(KW))$ and partial decryption data Q_{CT} to DU.

$Decrypt(PK,CT,Q_{CT},SK_{TA})$: Upon receiving the Q_{CT} and the CT from D-CSP, DU can completely decrypt the ciphertext and obtain the message $M=\frac{C_0\cdot e(d_{\theta 1},C_\theta)}{Q_{CT}\cdot e(C_1,d_{\theta 0})}$.

Correctness. The proposed KSF-OABE construction is correct as the following equations hold.

$$Q_{CT}=\frac{\prod_{i\in S} e(C_1,I_{i0})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(I_{i1},C_i)^{\Delta_{i,S}(0)}}=\frac{\prod_{i\in S} e(g^s,g_2^{q(i)}(g_1h_i)^{r_i})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(g^{r_i},(g_1h_i)^s)^{\Delta_{i,S}(0)}}$$

$$=\frac{e(g,g_2)^{s\sum_{i\in S}q(i)\Delta_{i,S}(0)}\prod_{i\in S} e(g^s,(g_1h_i)^{r_i})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(g^{r_i},(g_1h_i)^s)^{\Delta_{i,S}(0)}}$$

$$=e(g,g_2)^{sx_1}$$

$$k_{kw}=\frac{e(K_1,T_q(kw))}{e(D_1,K_2)}=\frac{e(g^s,H_1(kw)QK^u)}{e({d_{\theta 1}}^u,(g_1h)^s)}$$

$$=\frac{e(g^s,H_1(kw)g_2^x(g_1h)^{r_\theta u})}{e(g^{r_\theta u},(g_1h)^s)}$$

$$=\frac{e(g^s,(g_1h)^{r_\theta u})e(g,H_1(kw))^s e(g,g_2)^{sx}}{e(g^{r_\theta u},(g_1h)^s)}$$

$$=e(g_1,g_2)^s\cdot e(g,H_1(kw))^s$$

$$M=\frac{C_0\cdot e(d_{\theta 1},C_\theta)}{Q_{CT}\cdot e(C_1,d_{\theta 0})}=\frac{Me(g_1,g_2)^s e(g^{r_\theta},(g_1h)^s)}{e(g,g_2)^{sx_1}e(g^s,g_2^{x_2}(g_1h)^{r_\theta})}$$

$$=\frac{Me(g_1,g_2)^s e(g^{r_\theta},(g_1h)^s)}{e(g_1,g_2)^s e(g^s,(g_1h)^{r_\theta})}$$

$$=M$$

5 Security Proof

The first challenge of our construction on security and privacy is to defend the conspiracy attack from dishonest users and D-CSP. The conspiracy attack can be resisted because the master key x is randomly divided. We assume that there are two users who submit request on the generation of private keys. The master key x are randomly divided into two parts twice, and we get the splits (x_1,x_2) and (x_1',x_2'). We have $x_1+x_2=x \bmod p$ and $x_1'+x_2'=x \bmod p$. We use x_1, x_1' to generate the outsourcing private key SK_{KGCSP}, SK_{KGCSP}'. x_2, x_2' are used to generate corresponding local private key SK_{TA}, SK_{TA}' respectively. If and only if SK_{KGCSP} matches SK_{TA}, or SK_{KGCSP}' matches SK_{TA}', the ciphertext will be fully decrypted. Although the dishonest users collude with S-CSP and D-CSP to get all SK_{KGCSP} of users, they still aren't able to forge a valid private key SK of any user. We consider the security against $Type-I$ adversary here.

Theorem 1. The proposed KSF-OABE scheme with keyword search function is secure against chosen-plaintext attack launched by $Type-I$ adversary in selective model under DBDH assumption.

Proof. Assume that A is an $Type-I$ adversary that can break the proposed scheme, we can build an algorithm B that uses A as a sub-algorithm to solve the DBDH problem as follows. $H_1(\cdot)$, $H_2(\cdot)$ are defined as random oracles.

$Setup$: Algorithm B receives a challenge attribute set ω^* from A. B sends ω^* to challenger C as its challenge attribute set. Challenger C gives B the public parameter $(X=g^x, Y=g^y, Z=g^z)$. Algorithm B sets $g_1=X$, $g_2=Y$ and $h=g_1^{-1}g^{-\alpha}$ where α is selected from Z_p randomly. For each $i\in\omega^*$, B selects $\alpha_i\in Z_p$ randomly and sets $h_i=g_1^{-1}g^{\alpha_i}$. For each $i\notin\omega^*$, B selects $\alpha_i\in Z_p$ randomly and sets $h_i=g^{\alpha_i}$. B also chooses two collision-resistant hash functions where $H_1:\{0,1\}^*\rightarrow G_1$ and $H_2:G_2\rightarrow\{0,1\}^{\log p}$. B sends the public parameter $(g,g_1,g_2,h,h_1,...,h_n,H_1,H_2)$ to A.

$Query$ phase 1: The algorithm B initializes an empty table T. The adversary A makes any of the following queries adaptively:

H_1,H_2-query. The adversary A can ask the random oracles H_1 or H_2 at any time. In order to answer H_1 queries, the algorithm B maintains a list $<kw_i,\beta_i,b_i,c_i>$ called H_1-list. The list is initially empty. When the adversary A asks the random oracle H_1 at the point of $kw_i \in \{0,1\}^*$, the algorithm B returns as follows:

a. If the kw_i has already existed in the H_1 -list $< kw_i, \beta_i, b_i, c_i >$, algorithm B responds with $H_1(kw_i) = \beta_i$ where $\beta_i \in G_1$.

b. Otherwise, algorithm B generates $c_i \in \{0,1\}$ by flipping a coin and picks a $b_i \in Z_p$ randomly. If $c_i = 0$, B computes $\beta_i \leftarrow g_2 g^{b_i} \in G_1$. Else, B computes $\beta_i \leftarrow g^{b_i} \in G_1$.

c. The algorithm B adds the tuple $< kw_i, \beta_i, b_i, c_i >$ to the H_1 -list and sets $H_1(kw_i) = \beta_i$. B sends β_i to A .

Similar to H_1 -queries, the adversary A can issue a query on H_2 at any time. In order to respond to the query $H_2(k_i)$ on k_i, the algorithm B maintains a list for tuple $< k_i, I_i >$ called H_2 -list. The list is also initially empty. If the query on k_i exists in H_2 -list. B responds I_i to A . Otherwise, B randomly picks $I_i \in \{0,1\}^{\log p}$ for every k_i and sets $H_2(k_i) = I_i$. Algorithm B adds the pair (k_i, I_i) to H_2 -list. B sends I_i to A .

$OABE-KeyGen_{out}$ query. Upon receiving the query of private key SK_{KGCSP} on A, B checks if the tuple $(A, SK_{KGCSP}, \cdot, \cdot, \cdot)$ exists in T. If so, B returns SK_{KGCSP} to A . Else, if $|A \cap \omega^*| < d$, the algorithm B picks $x_2 \in Z_p$ randomly and defines three sets Γ, Γ' and S, where $\Gamma = A \cap \omega^*$, $|\Gamma'| = d-1$, $\Gamma \subseteq \Gamma' \subseteq A$ and $S = \Gamma' \cup \{0\}$. Then, for each $i \in \Gamma'$, B computes $d_{i0} = g_2^{\tau_i}(g_1 h_i)^{r_i}$ and $d_{i1} = g^{r_i}$ where τ_i, $r_i \in Z_p$ are selected at random. For each $i \in A \setminus \Gamma'$, We have $r_i = -y\Delta_{0,S}(i) + r_i'$ implicitly. B sets $d_{i0} = g_2^{\sum_{j\in\Gamma'} \Delta_{j,S}(i)\tau_j - (x_2+\alpha_i)\Delta_{0,S}(i)}(g_1 h_i)^{r_i'}$ and $d_{i1} = g_2^{-\Delta_{0,S}(i)} g^{r_i'}$ by choosing $r_i' \in Z_p$ randomly. B stores the partial private key in table T. Otherwise, B outputs $\perp$ and terminates the game.

$OABE-KeyGen_{in}$ query. Upon receiving a private key query for A with $|A \cap \omega^*| < d$, the algorithm B checks if the tuple $(A, \cdot, SK, \cdot, \cdot)$ exists in T. If so, B returns SK to A . Else, if x_2 for such tuple has not been selected in $OABE-KeyGen_{out}$ query, B picks $x_2 \in Z_p$ randomly and obtains SK_{KGCSP} which is similar to $OABE-KeyGen_{out}$ query (i.e. $|A \cap \omega^*| < d$). Algorithm B computes $SK_{TA} = \{d_{\theta 0} = g_2^{x_2}(g_1 h)^{r_\theta}, d_{\theta 1} = g^{r_\theta}\}$, where $r_\theta \in Z_p$ is selected at random. B adds $(A, SK_{KGCSP}, SK, \cdot, \cdot)$ into T and returns $SK = (SK_{KGCSP}, SK_{TA})$ to A .

QuiryKey query. Upon receiving a query private key request on A and a commitment value $C = g_2^{1/u_A}$, algorithm B checks whether the tuple $(A, \cdot, \cdot, QK, C)$ exists in T. If so, B returns QK to A . Otherwise B sends $C = g_2^{1/u_A}$ to challenger C , C responds B with g_2^{x/u_A}. B sets $QK = g_2^{x/u_A}(g_1 h)^{r_\theta}$ and sends it to A .

Trapdoor query. The adversary A makes a query for the trapdoor with respect to the keywords kw_i with an access policy A. If the access policy A does not exist in table T, algorithm B generates SK_{KGCSP} and SK according to $OABE-KeyGen_{out}$ query and $OABE-KeyGen_{in}$ query. Otherwise, the algorithm B responds as follows:

a. The algorithm B sends g_2 to challenger C , C responds B with g_2^x.

b. The algorithm B runs the above algorithm H_1 -queries to acquire an $\beta_i \in G_1$ and sets $H_1(kw_i) = \beta_i$. Let $< kw_i, \beta_i, b_i, c_i >$ be the tuple in the H_1 -list. If c_i=0 , B responds failure and terminates the game.

c. Else, we have $c_i = 1$ and $\beta_i = g^{b_i} \in G_1$. Algorithm B selects a u from Z_p randomly, and search the table T for the SK and sets $T_q(kw) = g^{b_i} g_2^x (g_1 h)^{r_\theta u} = g^{b_i} QK^u$, $I = (I_{i0} = d_{i0}, I_{i1} = d_{i1})$ for all $i \in A$, $D_1 = d_{\theta 1}{}^u$. Therefore $T_{kw} = (T_q(kw), I, D_1)$ is a valid trapdoor for the keyword kw. Algorithm B returns T_{kw} to A .

Challenge: The adversary A sends two messages M_0, M_1 with equal-length and an attribute set ω^* to B for challenging. At the same time, A generates a pair of keywords kw_0 and kw_1 that he wants to challenge. B responds as follows:

(1) The algorithm randomly chooses the M_υ, where $\upsilon \in \{0,1\}$. B encrypts M_υ as $CT^* = (\omega^* \cup \{\theta\}, M_\upsilon T, g^z, g^{-z\alpha}, \{g^{z\alpha_i}\}_{i\in\omega^*})$. If $\mu = 0$, algorithm B sets $T = e(g,g)^{xyz}$. Let $s = z$ and $C_0 = M_\upsilon T = M_\upsilon e(g,g)^{xyz} = M_\upsilon g_1, g_2)^z e(g_1, g_2)^z$, $C_1 = g^z$, $C_i = g^{z\alpha_i} = (g_1 g_1^{-1} g^{\alpha_i})^z = (g_1 h_i)^z$ or $i \in \omega^*$, and ω^*. Otherwise, T is selected from G_2 at random. Therefore, the encryption process of M_υ is random and the attribute set ω^* is embedded in the ciphertext.

(2) The algorithm B runs the above H_1 -queries algorithm twice to obtain $\beta_0, \beta_1 \in G_1$. B sets $H_1(kw_0) = \beta_0$ and $H_1(kw_1) = \beta_1$. For $i \in \{0,1\}$, B stores the corresponding tuples $< kw_i, \beta_i, b_i, c_i >$ in the H_1 -list. If both $c_i = 1$, B responds failure and terminates the game.

(3) Now we have at lowest one of c_0, c_1 is equal to 0. The algorithm B picks a $\varphi \in \{0,1\}$ randomly and sets $c_\varphi = 0$.

(4) The algorithm B sets $IX(kw_\varphi) = (K_1, K_2, K_\varphi)$ where $K_1 = g^z$, $K_2 = g^{-z\alpha} = (g_1 g_1^{-1} g^{-\alpha})^z = (g_1 h)^z$. We let $k_\varphi = e(g_1, g_2)^z \cdot e(g, g_2 g^{b_\varphi})^z$ and B runs the above H_2 -queries algorithm to obtain $K_\varphi = J \in \{0,1\}^{\log p}$. The

algorithm B stores the tuple $<k_\varphi, J>$ in the H_2 -list. B responds to A with the challenge $IX(kw_\varphi)=(K_1,K_2,J)$ and CT^*.

Note that this challenge index implicitly defines :

$$J=H_2(e(g_1,g_2)^z\cdot e(g,H_1(kw_\varphi))^z)$$
$$=H_2(e(g_1,g_2)^z\cdot e(g,g_2g^{b_\varphi})^z)$$

With this definition, $IX(KW)$ is a valid index for kw_φ.

Query Phase 2. The adversary A adaptively issues H_1,H_2 -query, $OABE-KeyGen_{out}$, $OABE-KeyGen_{\text{in}}$, $Trapdoor$ queries, as $Query$ Phase 1 with the restrictions as follows :

(1) The adversary A can not launch the $OABE-KeyGen_{out}$ and $OABE-KeyGen_{\text{in}}$ query that would result in an access policy A which satisfies the attribute set ω^* and is added to T.

(2) The adversary A can not issue the $Trapdoor$ query on kw_i that kw_i equals to neither kw_0 nor kw_1.

Guess. The adversary A gives a guess υ' for υ. If $\upsilon'=\upsilon$,B outputs $\mu'=0$ to show that it is given a DBDH-tuple. Else, it outputs $\mu'=1$ to show that it is given a random 4-tuple. Also the adversary A outputs its guess $\varphi'\in\{0,1\}$ to indicate whether the challenge $IX(kw_\varphi)$ is the result of $IX(kw_0)$ or $IX(kw_1)$. A computes as follow:

$$k_\varphi=\frac{e(K_1,T_q(kw))}{e(D_1,K_2)}$$
$$=\frac{e(g^s,H_1(kw)QK^u)}{e(d_{\theta1}{}^u,(g_1h)^s)}=\frac{e(g^s,g^{b_i}g_2^x(g_1h)^{r_\theta u_\text{A}})}{e(g^{r_\theta u},(g_1h)^s)}$$
$$=\frac{e(g,g_2)^{sx}e(g,g^{b_i})^s e(g,g_1h)^{sr_\theta u_\text{A}}}{e(g,g_1h)^{sr_\theta u}}$$
$$=\frac{e(g_1,g_2)^{s}e(g,g^{b_i})^s e(g,g_1h)^{sr_\theta u_\text{A}}}{e(g,g_1h)^{sr_\theta u}}$$

If A can break our proposed scheme. We have $k_\varphi=e(g_1,g_2)^s e(g,g^{b_\varphi})^s$. At the same time, the algorithm B searches $IX(kw_\varphi)$ from H_2 -list for k_φ and outputs $k_\varphi/e(K_1,g_2g^{b_\varphi})=e(g,g)^{xyz}$.

The second challenge of our construction on security and privacy is to defend the conspiracy attack from curious KG-CSP. Apparently, in order to recover the message, the $Type-II$ adversary must obtain $e(g_1,g_2)^s$. However, with the help of Lagrange interpolation method and CT, the adversary can recover $e(g,g_2)^{sx_1}$. Because of the use of blinding factor, the adversary cannot get $e(g,g_2)^{sx_2}$ unless the obtain of binding factor BF and SK_{TA}.

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

Theorem 2. The proposed KSF-OABE scheme with keyword search function is secure against chosen-plaintext attack launched by $Type-II$ adversary.

Proof. Suppose that A is a $Type-II$ adversary that can break our scheme, we are able to build an algorithm B that can use A as a sub-routine to solve the DBDH problem as follows. $H_1(\cdot)$, $H_2(\cdot)$ are defined as random oracles.

$Setup$: The algorithm B receives a challenge attribute set ω^* from A . B sends ω^* to the challenger C as its challenge attribute set. The challenger C gives B the public parameter $(X=g^x,Y=g^y,Z=g^z)$. The algorithm B sets $g_1=X$, $g_2=Y$ and $h=g_1^{-1}g^{-\alpha}$ where α is selected from Z_p randomly. For each $i\in\omega^*$, B selects $\alpha_i\in Z_p$ randomly and sets $h_i=g_1^{-1}g^{\alpha_i}$. B also chooses two collision-resistant hash functions where $H_1:\{0,1\}^*\to G_1$ and $H_2:G_2\to\{0,1\}^{\log p}$. B sends the public parameter $(g,g_1,g_2,h,h_1,...,h_n,H_1,H_2)$ to A .

$Query$ phase 1: The algorithm B initializes T as an empty table. The adversary A makes any of the following queries adaptively:

H_1,H_2 -query. This step is the same as the theorem 1.

$OABE-KeyGen_{out}$ query. Upon receiving the query of private key SK_{KGCSP} on A , B checks if the tuple $(A,SK_{KGCSP},\cdot,\cdot,\cdot)$ exists in table T. If so, B returns SK_{KGCSP} to A . Else, B picks $x_1\in Z_p$ randomly, for each $i\in A$, B computes $d_{i0}=g_2^{x_1}(g_1h_i)^{r_i}$ and $d_{i1}=g^{r_i}$ where $r_i\in Z_p$ is selected at random. B stores the partial private key in table T. Otherwise, B outputs $\perp$ and terminates the game.

$OABE-KeyGen_{\text{in}}$ query. On receiving a private key query for A , the algorithm B checks if the tuple $(A,\cdot,SK,\cdot,\cdot)$ exists in table T. If so, B returns SK to A . Else, for each A, B computes $d_{\theta0}=g_2^{\tau}(g_1h)^{r_\theta}$ and $d_{\theta1}=g^{r_\theta}$ where τ , $r_\theta\in Z_p$ are selected at random. B B adds $(A,SK_{KGCSP},SK,\cdot,\cdot)$ to T and returns $SK=(SK_{KGCSP},SK_{TA})$ to A .

$QuiryKey$ query. This step is the same as the theorem 1.

$Trapdoor$ query. This step is the same as the theorem 1.

Challenge: This step is the same to the theorem 1.

Query Phase 2. The adversary A continues to adaptively issue H_1,H_2 -query, $OABE-KeyGen_{out}$, $OABE-KeyGen_{\text{in}}$, $Trapdoor$ queries as $Query$ Phase 1. The restrictions are as follows.

(1) The adversary A can not launch the $OABE-KeyGen_{out}$ and $OABE-KeyGen_{\text{in}}$ query that would result in an access policy A which satisfies the attribute set ω^* and was added to T.

(2) The adversary A can not issue the $Trapdoor$ query on kw_i that kw_i equals to neither kw_0 nor kw_1.

Guess. The adversary A gives a guess υ' for υ. If $\upsilon'=\upsilon$,B outputs $\mu'=0$ to show that it is given a DBDH-tuple. Else, it outputs $\mu'=1$ to show it is given a random 4-tuple. Also the adversary A outputs its guess $\varphi'\in\{0,1\}$ to indicate whether the challenge $IX(kw_\varphi)$ is the result of $IX(kw_0)$ or $IX(kw_1)$. A computes as follow:

$$
\begin{aligned}
k_\varphi &= \frac{e(K_1, T_q(kw))}{e(D_1, K_2)} \\
&= \frac{e(g^s, H_1(kw)QK^u)}{e(d_{\theta 1}{}^u, (g_1h)^s)} = \frac{e(g^s, g^{b_i} g_2^x (g_1h)^{r_\theta u_A})}{e(g^{r_\theta u}, (g_1h)^s)} \\
&= \frac{e(g, g_2)^{sx} e(g, g^{b_i})^s e(g, g_1h)^{sr_\theta u_A}}{e(g, g_1h)^{sr_\theta u}} \\
&= \frac{e(g_1, g_2)^s e(g, g^{b_i})^s e(g, g_1h)^{sr_\theta u_A}}{e(g, g_1h)^{sr_\theta u}}
\end{aligned}
$$

If A can break the proposed scheme. We have $k_\varphi = e(g_1, g_2)^s e(g, g^{b_\varphi})^s$. At the same time, the algorithm B searches $IX(kw_\varphi)$ from H_2 -list for k_φ and outputs $k_\varphi / e(K_1, g_2 g^{b_\varphi}) = e(g,g)^{xyz}$.

6 Performance Analysis

6.1 Complexity Analysis

In Table2 and Table 3, we briefly compare our scheme with the work in [16, 17, 27]. *PK*, *MK*, *SK*, *CT*, *TK*, *RK*, *CT*' represent the size of public key, master key, private key, ciphertext length, transform key, retrieving key and transformed ciphertext excluding the access structure respectively. Additionally, *Encrypt*, *Transform*, $Decrypt_{out}$, *Decrypt* denote the computational costs of the algorithms encryption, transformation, outsourcing decryption, decryption respectively. $|G_1|$, $|G_T|$, $|Z_P|$ denote the bit-length of the elements belong to G_1, G_T, Z_P. kG_1, kC_e, kH denote the k -times calculation over the group G_1, pairing and hash function. Let $U = \{att_1, ..., att_n\}$ be the attribute universe. N_1 and N_2 are amount of the attributes associated with ciphertext and private key respectively. K is the number of keywords associated with a ciphertext. As the operation cost over Z_P is much less than group and pairing operation, we ignore the computational time over Z_P.

TABLE 2 Size of each value

	PK	MK	SK	CT	TK	RK	CT’
LCL 13 [16]	$(n+4)\|G_1\|$	$\|Z_P\|$	$2N_2\|G_1\|$	$(N_1+2)\|G_1\|+\|G_T\|$	$(2N_2-1)\|G_1\|$	$2\|G_1\|$	$2\|G_1\|+2\|G_T\|$
LHL 14 [17]	$(n+4)\|G_1\|$	$\|Z_P\|$	$2N_2\|G_1\|+\|Z_P\|$	$(N_1+2)\|G_1\|+\|G_T\|$	$2N_2\|G_1\|$	$\|Z_P\|$	$\|G_T\|$
GHW 13[27]	$4\|G_1\|$	$\|Z_P\|$	none	$(N_1+1)\|G_1\|+\|G_T\|+n\|Z_P\|$	$2N_2\|G_1\|$	$\|Z_P\|$	$2\|G_T\|$
Our scheme	$(n+4)\|G_1\|$	$\|Z_P\|$	$2N_2\|G_1\|$	$(N_1+4)\|G_1\|+(1+K)\|G_T\|$	$2N_2\|G_1\|$	$\|Z_P\|$	$2\|G_T\|$

TABLE 3 Computational cost

	Encrypt	*Transform*	$Decrypt_{Out}$	*Decrypt*
LCL 13 [16]	$C_e+2G_T+(3+2N_1)G_1$	none	$2N_1C_e+(2N_1+1)G_T$	$2C_e+3G_T$
LHL 14 [17]	$C_e+2G_T+(3+2N_1)G_1$	$2N_2G_1$	$2(N_1+1)C_e+(2N_1+3)G_T$	$3G_T$
GHW 13 [27]	$C_e+(N_1+1)G_1+3G_T+N_1H$	$2N_2G_1$	$(N_1+1)C_e+2N_1G_1+N_1G_T$	$2G_T$
Our scheme	$(1+K)C_e+2(1+K)G_T+(3+2N_1)G_1+KH$	$4G_1$	$2(N_1+1)(C_e+G_T)$	$2C_e+3G_T$

6.2 Efficiency Analysis

We compared the performance of the four stages in our scheme with the scheme [16] as figures below. Our experiment is simulated with the java pairing-based cryptography (JPBC) library version 2.0.0 [35], which is a port of the pairing-based cryptography (PBC) library [36] in C. When selecting a secure elliptic curve, two factors should be considered: the group size *l* of the elliptic curve and the embedding degree *d*. To achieve the 1024-bit RSA security, these two factors should satisfy $l \times d \geqslant 1024$. We implement our scheme on Type *A* curve $y^2 = x^3 + x$ where *p* is 160 bits, *l* = 512. We select SHA− as the hash function. We implement our scheme and the scheme [16] on a Windows machine with Intel Core 2 processor running at 2.13 GHz and 4G memory. The running environment of our experiment is Java Runtime Environment 1.7 (JRE1.7), and the Java Virtual Machine (JVM) used to compile our programming is 32 bit (x86) which brings into correspondence with our operation system.

For simplicity, we assume that DU submits one keyword and obtains one partial decryption data to be decrypted fully in our system. From Fig. 2(a) and Fig. 2(c), we see that the computation costs at the stages of Setup and Encryption grow linearly with the amount of the attribute in both systems and the computation costs in our scheme which is similar to the scheme [16]. Fig. 2(b) shows that the computation cost at the stage of KeyGen for KG-CSP grows linearly with the amount of the attributes in the system, but the computational cost for TA just keeps in a low level. The computation costs in our scheme are similar to the scheme [16] on both TA and KG-CSP side. Fig. 2(d) shows that the computation cost at the stage of Decryption for DU grows linearly with the amount of data belong to the DU in the system for scheme [16], but the computational cost in our system keeps in a low level.

Field Code Changed

Formatted: Font color: Auto

Formatted: Font color: Auto

Formatted: Font color: Auto

Formatted: Font color: Auto

Formatted: Font color: Auto

Formatted: Font color: Auto

Field Code Changed

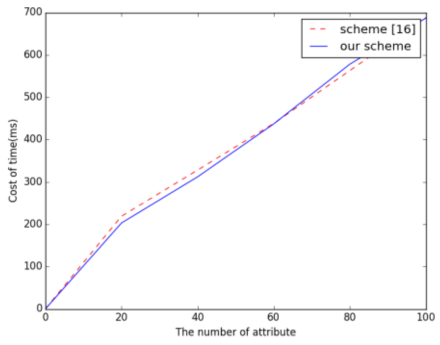

Fig.2(a) Setup

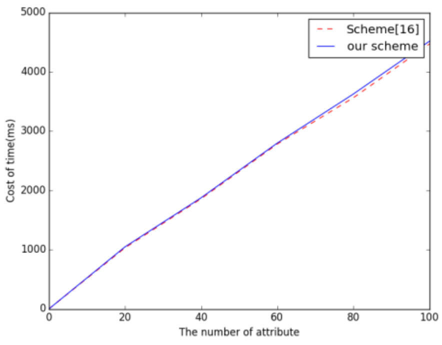

Fig.2(c) Encryption

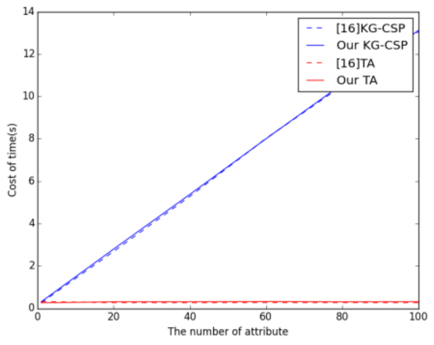

Fig.2(b) KeyGen

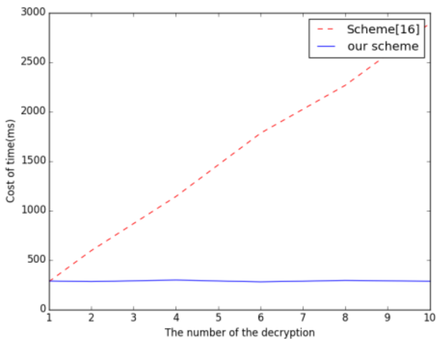

Fig.2(d) Decryption

7 Conclusion

In this article, we propose a CP-ABE scheme that provides outsourcing key-issuing, decryption and keyword search function. Our scheme is efficient since we only need to download the partial decryption ciphertext corresponding to a specific keyword. In our scheme, the time-consuming pairing operation can be outsourced to the cloud service provider, while the slight operations can be done by users. Thus, the computation cost at both users and trusted authority sides is minimized. Furthermore, the proposed scheme supports the function of keywords search which can greatly improve communication efficiency and further protect the security and privacy of users. Actually, we are easy to extend our KSF-OABE scheme to support access structure represented by tree in [9].

8 Future Work

Verifiability is an important feature of KSF-OABE, so one of our future works is to construct KSF-OABE which can provide verifiability. Furthermore, our scheme was only RCCA secure in the random oracle model, hence constructing KSF-OABE which is CCA secure in the standard model is another future work.

Acknowledgment

We would like to thank anonymous referees for their valuable comments and suggestions to improve our paper. This work was supported in part by the National Natural Science Foundation of China (61272542, 61300213), the Fundamental Research Funds for the Central Universities (2013B07014), the Priority Academic Program Development of Jiangsu Higher Education Institutions and Six Talent Peaks Project of Jiangsu Province (2015-DZXX-020)

REFERENCES

[1] S. Pearson, Y. Shen and M. Mowbray, "A Privacy Manager for Cloud Computing," *Proc. First International Conference Cloud Computing (CloudCom '09)*, M. Gilje-Jaatun, G. Zhao and C. Rong, eds., LNCS 5931, Berlin: Springer-Verlag, pp. 90-106, 2009.

[2] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *EUROCRYPT '05*, LNCS, vol. 3494, pp. 457-473, 2005.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proc. 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89-98, 2006, doi:10.1145/ 1180405.1180418.

[4] J.W. Li, C.F. Jia, J. Li and X.F. Chen, "Outsourcing Encryption of Attribute-Based Encryption with Mapreduce," *Proc. 14th International Conference on Information and Communications Security (ICICS '12)*, LNCS 7618, Berlin: Springer-Verlag, pp. 191-201, 2012. doi: 10.1007/ 978-3-642-34129-8_17

[5] A. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters, "Attribute-Based Encryption and (Hierarchical) Inner Product Encryption," *EUROCRYPT '10*, H. Gilbert, ed., LNCS 6110, Berlin: Springer-Verlag, pp. 62-91, 2010.

[6] J.G. Han, W. Susilo, Y. Mu and J. Yan, "Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no.11, pp. 2150-2162, Nov. 2012, doi: 10.1109/ TPDS.2012.50.

[7] T. Okamoto and K. Takashima, "Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption," *CRYPTO '10*, T. Rabin, ed., LNCS 6223, Berlin: Springer-Verlag, pp. 191-208, 2010.

[8] W.R. Liu, J.W. Liu, Q.H. Wu, B. Qin, and Y.Y. Zhou, "Practical Direct Chosen Ciphertext Secure Key-Policy Attribute-Based Encryption with Public Ciphertext Test," *ESORICS '14*, LNCS 8713, Berlin: Springer-Verlag, pp. 91-108, 2014.

[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy Attribute-Based Encryption," *Proc. IEEE Symp. Security and Privacy (SP '07)*, pp. 321-334, May. 2007, doi:10.1109/ SP.2007.11.

[10] L. Cheung and C. Newport, "Provably Secure Ciphertext Policy ABE," *Proc. 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 456-465, 2007, doi:10.1145/ 1180405.1180418.

[11] H.L. Qian, J.G. Li and Y.C. Zhang, "Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption with Fully Hidden Access Structure," *Proc. 15th International Conference on Information and Communications Security (ICICS '13)*, LNCS 8233, Berlin: Springer-Verlag, pp. 363-372, 2013.

[12] H. Deng, Q.H. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J.W. Liu, and W.C. Shi, "Ciphertext-Policy Hierarchical Attribute-Based Encryption with Short Ciphertexts," *Information Sciences*, vol. 275, no. 8, pp. 370-384, Aug. 2014.

[13] J.T. Ning, Z.F. Cao, X.L. Dong, L.F. Wei and X.D. Lin, "Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability," *ESORICS '14*, LNCS 8713, Berlin: Springer-Verlag, pp. 55-72, 2014.

[14] Z. Liu, Z.F. Cao, and D. S. Wong, "Traceable CP-ABE: How to Trace Decryption Devices Found in the Wild," *IEEE Transactions on Information Forensics and Security*, vol. 10, no.1, pp. 55-68, Jan. 2015.

[15] J.T. Ning, X.L. Dong, Z.F. Cao, L.F. Wei and X.D. Lin, "White-Box Traceable Cphertext-Policy Attribute-Based Encryption Supporting Flexible Attributes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no.6, pp. 1274-1288, Jun. 2015.

[16] J. Li, X.F. Chen, J.W. Li, C.F. Jia, J.F. Ma and W.J. Lou, "Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption," *Proc. 18th European Symposium on Research in Computer Security (ESORICS '13)*, LNCS 8134, Berlin: Springer-Verlag, pp. 592-609, 2013.

[17] J. Li, X. Huang, J. Li and X. Chen, "Securely Outsourcing Attribute-Based Encryption with Checkability," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201-2210, Oct 2013/ Jul 2014, doi:10.1109/ TPDS.2013.271.

[18] S. Hohenberger and A. Lysyanskaya, "How to Securely Outsource Cryptographic Computations," *Proc. Second Theory of Cryptography Conference (TCC'05)*, J. Kilian, ed., LNCS 3378, Berlin: Springer-Verlag, pp. 264-282, 2005.

[19] M. Yang, F. Liu, J.L. Han and Z.L. Wang, "An Efficient Attribute Based Encryption Scheme with Revocation for Outsourced Data Sharing Control," *Proc. 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC '11)*, pp. 516-520, Oct. 2011, doi:10.1109/ IMCCC.2011.134.

[20] J.Z. Lai, R.H. Deng and C. Guan, "Attribute-Based Encryption with Verifiable Outsourced Decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343-1354, 2013, doi: 10.1109/ TIFS.2013.2271848.

[21] F. Zhao, T. Nishide and K. Sakurai, "Realizing Fine-Grained and Flexible Access Control to Outsourced Data with Attribute-Based Cryptosystems," *Proc. 7th International Conference. Information Security Practice and Experience (ISPEC '11)*, F. Bao and J. Weng, eds., LNCS 6672, Berlin: Springer-Verlag, pp. 83-97, 2011.

[22] X. Chen, J. Li, X. Huang and J. Li, "Secure Outsourced Attribute-Based Signatures," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3285-3294, Jan/ Nov 2014, doi:10.1109/ TPDS.2013.2295809.

[23] G. Ateniese, K. Fu, M. Green and S. Hohenberger, "Improved Proxy Re-encryption Scheme with Application to Secure Distributed Storage," *J. ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1-30, Feb. 2006, doi:10.1145/1127345.1127346.

[24] B. Libert and D. Vergnaud, "Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption," *Proc. 11th International Workshop on Practice and Theory in Public Key Cryptography (PKC '08)*, R. Cramer, ed., LNCS 4939, Berlin: Springer-Verlag, pp. 360-379, 2008.

[25] J. Hur and D.K. Noh, "Attribute-based Access Control with Efficient Revocation in Data Outsourcing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no.7, pp. 1214-1221, Nov 2010, doi: 10.1109/ TPDS.2010.203.

[26] 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 Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131-143, Jan. 2012, doi:10.1109/ TPDS.2012.97.

[27] M. Green, S. Hohenberger and B. Waters, "Outsourcing the Decryption of ABE Ciphertexts," *Proc. 20th USENIX Conference on Security (SEC '11)*, pp. 34, 2011.

[28] D. Boneh, G.D. Cirescenzo, R. Ostrovsky and G. Persiano, "Public Key Encryption with Keyword Search," *EUROCRYPT '04* , C. Cachin and J.L. Camenisch, eds., LNCS 3027, Berlin: Springer-Verlag, pp. 506-522, 2004.

[29] P. Xu, H. Jin, Q.H. Wu, and W. Wang, "Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack," *IEEE Transactions on Computers*, vol. 62, no.11,

Field Code Changed

pp. 2266-2277, Nov. 2013.

[30] J.G. Li, Y.R. Shi, and Y.C. Zhang, "Searchable Ciphertext-Policy Attribute-Based Encryption with Revocation in Cloud Storage," *International Journal of Communication Systems*, 2015, doi: 10.1002/dac.2942

[31] Q.J. Zheng, S.H. Xu, G. Ateniese, "VABKS: Verifiable Attribute-Based Keyword Search over Outsourced Encrypted Data," *INFOCOM* '14, pp. 522-530, 2014, doi: 10.1109/INFOCOM.2014.6847976.

[32] H.L. Qian, J.G. Li, Y.C. Zhang and J.G. Han, "Privacy Preserving Personal Health Record Using Multi-Authority Attribute-Based Encryption with Revocation," *International Journal of Information Security*, pp. 1-11, 2015, doi: 10.1007/s10207-014-0270-9.

[33] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp.612-613, Nov 1979.

[34] R. Canetti, H. Krawczyk and J.B. Nielsen, "Relaxing Chosen-Ciphertext Security," CRYPTO '03, D. Boneh, ed., LNCS 2729, Berlin: Springer-Verlag, pp. 565-582, 2003.

[35] A.D. Caro, "Java Pairing-Based Cryptography Library," http://gas.dia.unisa.it/projects/jpbc, 2013.

[36] B. Lynn, "Pairing-Based Cryptography (PBC) Library," http://crypto.stanford.edu/pbc, 2013.

Formatted: Font color: Auto

Formatted: Font color: Auto

Formatted: Font color: Auto