INTEGRITY VERIFICATION AND ATTRIBUTE BASED ENCRYPTION FOR CLOUD STORAGE

V. Venkata Surya Deep¹, M. Nagaraju²

¹Student, Dept. of Computer Science Engineering, BENAIH Institute of Technology and Science, Burugupudi, Rajamahendravaram, Andhra Pradesh, India.

²Associate Professor, Dept. of Computer Science Engineering, BENAIH Institute of Technology and Science, Burugupudi, Rajamahendravaram, Andhra Pradesh, India.

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 the cloud. This kind of computing model brings challenges to the security and privacy of data stored in the cloud. Attribute-based encryption (ABE) technology has been used to design a fine-grained access control system, which provides one good method to solve the security issues in the 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 the cloud by outsourcing the heavy computation to cloud service provider (CSP). However, as the amount of encrypted files stored in the cloud is becoming very huge, which will hinder efficient query processing. To deal with the above problem, we present a new cryptographic primitive called attribute-based encryption scheme. The proposed ABE 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 the trapdoor.

Keywords - attribute-based encryption, cloud computing, integrity verification, outsourced key-issuing, outsourced decryption, cloud storage.

1. INTRODUCTION

CLOUD computing is a new computation model in which computing resources are regarded as a 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. 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 mechanism to protect encrypted sensitive information. Sahai and Waters 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 ciphertext or private keys. Two types of ABE schemes, namely key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE) 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 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 collision attack and avoid sensitive information leakage from access structure, Qian et al. Proposed a privacy-preserving decentralized ABE scheme with fully hidden access structure. In CP-ABE scheme, a malicious user maybe shares his attributes with other users, which might leak his decryption privilege as a decryption black box due to financial profits. In order to solve above problem, Cao et al. 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 draw backs 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 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. However, this scheme cannot support fine-grained access control on encrypted files. Some schemes have been proposed to focus on the above problems. Qian et al. 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, 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 integrates outsourced scheme with PEKS and present a novel cryptographic paradigm called outsourced attribute-based encryption 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 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.

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 outsourcing decryption are presented in Section 4. We evaluate our construction in Section 5. Finally, we draw our conclusion in Section 6.

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

Table 1 lists some notations utilized in this paper.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>Trusted Authority</td>
</tr>
<tr>
<td>KG-CSP</td>
<td>key generation cloud server provider</td>
</tr>
<tr>
<td>D-CSP</td>
<td>decryption cloud server provider</td>
</tr>
<tr>
<td>S-CSP</td>
<td>storage cloud server provider</td>
</tr>
<tr>
<td>DO</td>
<td>data owner</td>
</tr>
<tr>
<td>DU</td>
<td>data user</td>
</tr>
</tbody>
</table>

2.2 Bilinear Pairing

Let $G_1$ and $G_2$ be multiplicative cyclic groups with prime order $p$. Suppose $g$ is a generator of $G_1$. $e : G_1 \times G_2 \rightarrow G_T$ 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 $a, b \in \mathbb{Z}_p^*$ 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

Definition 1 (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 $X \in G_2$, $Y \in G_2$, $Z = g^r$, $x, y, z$ are selected from $\mathbb{Z}_p$ randomly and $T$ is selected from $G_2$ randomly. It’s difficult to decide whether $T = e(g, g)^{xy}$.

2.4 Access Structures

Definition 2 (Access Structure).

Suppose $\{ P_1, \ldots, P_n \}$ are a set of parties. A collection $A \subseteq \{ P_1, \ldots, P_n \}$ is monotone if $\mathbb{B}C \subseteq B_2C \subseteq \mathbb{B}_1C$ and $B_2C \subseteq \mathbb{B}_1C$ then $\mathbb{B}_2C$. A monotone access structure is a monotone collection $A$ which is a nonempty subset for $\{ P_1, \ldots, P_n \}$. The set in $A$ is called an authorized set, and the set out of $A$ is called an unauthorized set. Let $\mathbb{B}$ and $A$ be an attribute set and access policy. A predicate $\mathbb{B}(\mathbb{R}, A)$ is defined as follows:
3. SYSTEM ARCHITECTURE, FORMAL DEFINITION AND SECURITY MODEL

3.1 System Architecture

The system architecture for 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 to 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 ciphertext and keyword search service on the partially decrypted ciphertext 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 ciphertext will be shared with intended receivers whose access structure will be satisfied by attribute set embedded in ciphertext, that is to say the predicate $\mathbb{K}(\mathcal{A}) \# 1$. 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 ciphertext with the help of trapdoor associated with appointed keyword. Data user is responsible for choosing keywords to create trapdoor, and decrypting data.

3.2 Formal Definition

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

**Setup**($\mathbb{K}$): TA runs the **Setup** algorithm, which takes a security parameter $\mathbb{K}$ as input. It outputs the master secret key $\text{MSK}$ and system public parameter $\text{PK}$. TA publishes the system public parameter $\text{PK}$ and keeps the $\text{MSK}$ secret. It is described as $\text{Setup}(\mathbb{K}) = (\text{PK}, \text{MSK})$.

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

**OABE** $(\text{KeyGen}_{\text{out}})(\mathcal{A}, \text{OK}_{\text{KGCSP}})$: This algorithm is performed by KG-CSP which takes an access policy $\mathcal{A}$ and $\text{OK}_{\text{KGCSP}}$ as input. It outputs outsourcing private key $\text{SK}_{\text{KGCSP}}$. KG-CSP returns TA the $\text{SK}_{\text{KGCSP}}$. It is described as $\text{OABE} \text{ KeyGen}_{\text{out}}(\mathcal{A}, \text{OK}_{\text{KGCSP}}) = \text{SK}_{\text{KGCSP}}$.

**OABE** $(\text{KeyGen}_{\text{in}})(\text{OK}_{\text{TA}})$: This algorithm is performed by TA which takes $\text{OK}_{\text{TA}}$ as input. It outputs local private key $\text{SK}_{\text{TA}}$. It is described as $\text{OABE} \text{ KeyGen}_{\text{in}}(\text{OK}_{\text{TA}}) = \text{SK}_{\text{TA}}$. TA then sets user private key as $\text{SK} = \text{SK}_{\text{KGCSP}} \text{SK}_{\text{TA}}$ and sends $\text{SK}$ to user.

**KSF** $(\text{KeyGen})(\text{PK}, \text{MSK}, \mathcal{A}, q_{BF})$: This algorithm is performed by TA which takes $\text{PK}, \text{MSK}$, an access policy $\mathcal{A}$ 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 $\text{KSF} \text{ KeyGen}(\text{PK}, \text{MSK}, \mathcal{A}, q_{BF}) = QK$.

**Encrypt** $(\text{PK}, \mathcal{M}, q_{BF})$: This algorithm is run by DO which takes the system public parameter $\text{PK}$, a message $\mathcal{M}$ and an attribute set $\mathbb{K}$ as input. It outputs a ciphertext $\text{CT}$. It is described as $\text{Encrypt}(\text{PK}, \mathcal{M}, q_{BF}) = \text{CT}$. 

Index (PK, CT, KW): This algorithm is performed by DO which takes the system public parameter PK, and a keyword set KW as input. It outputs a searchable index of KW written as IX (KW). It is described as Index (PK, CT, KW)→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 Tkw corresponding to the keyword kw. It is described as Trapdoor (PK, QK, BF, kw)→Tkw.

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

Decrypt(PK, CT, QCT, SKTA): It is run by DU which takes the system public parameter PK, the searched ciphertext CT, the partial decryption ciphertext QCT, and the local private key of DU as input. It outputs the plaintext M for the DU. It is described as Decrypt (PK, CT, QCT, SKTA)→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 DCSP 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 SKGCSP and the trapdoor kw T 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 OKGCSP of any user.

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

We adopt a relaxation according to the secure notion called replayable CCA (RCCA) security in "R. Canetti, H. Krawczyk and J.B. Nielsen, Relaxing Chosen-Ciphertext Security", 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 "R. Canetti, H. Krawczyk and J.B. Nielsen" 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 A and keeps MSK secret.

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

1. OABE - KeyGenKey query. On input an access policy A, the challenger searches the tuple (A, SKGCSP, SK, T) in table T. If the tuple exists, it returns the outsourcing private key KGSK SK generated by KG-CSP. Otherwise, it runs the OABE - KeyGenKey (A, OKGCSP) algorithm to get SKGCSP. The challenger stores the outsourcing private key SKGCSP in table T and returns it to the adversary.

2. OABE - KeyGenKey query. According to an access policy A, the challenger searches the tuple (A, SKGCSP, SK, T) in table T. If the tuple exists, it returns the private key SK. Else, it runs the algorithm OABE - KeyGenKey (A, OKGCSP) to get the SKGCSP and the OABE - KeyGenKey (OKTA) algorithm to get local private key SKTA. The challenger sets SK = (SKGCSP, SKTA) 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, SKGCSP, SK, T) in table T. If the tuple exists, it returns the trapdoor key T 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 - KeyGenKey (PK, MSK, A, q), and SKGCSP to get SKGCSP and SKTA. The challenger sets SK = (SKGCSP, SKTA) and stores the private key SK in table T and returns it to the adversary.

4. Decrypt query. On input an access policy A and ciphertext (CT, QCT), the challenger queries the tuple (A, SKGCSP, SK, T) in table T. If the tuple exists, it implements Decrypt (PK, CT, QCT, SK) and returns M to the adversary. Else, it returns .

Challenge: The adversary sends two messages M0, M1 with equal-length and a challenge attribute set to the challenger, subject to the restriction that, can not satisfy A. The challenger chooses , and runs
Encrypt(PK,M,g) \rightarrow 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 $\mathbb{E}^*$.

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 $\mathbb{H}(\mathbb{E}^*)$ for $\mathbb{E}^*$ The advantage which the adversary can win the game is defined as $| Pr(\mathbb{H} = \mathbb{E}^*) - 1/2 |$. The game associated with Typell 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

$| Pr(\mathbb{H} = \mathbb{E}^*) - 1/2 |$.

 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 $\mathbb{E}^*$ prior to seeing the public parameters.

4. OABE SCHEME

Our scheme is based on the OABE proposed in [4]. We use tree-based access structure described as in [4]. $A$ is a tree-based access policy bound up with user private key, $\mathbb{E}^*$ is an attribute set embedded in ciphertext, $U$ is the attribute universe, and $d$ is a threshold value set in advance. If $\mathbb{H}(\mathbb{E},A) \equiv 1, S$ is an attribute set which satisfies $S \in \{\mathbb{E},A\}$ |

Setup ($\mathbb{E}$): 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 $e: G_1 \times G_2 \rightarrow G_T$ and defines the attributes in $U$ as values in $Z_p$. For simplicity, we set $n \equiv \lceil \log_2 |U| \rceil$ and take the first $n$ values in $Z_p$ to be the attribute universe. TA randomly selects an integer $i \equiv \lceil \log_2 n \rceil$, computes $g_i \equiv g^i$, and chooses $g_0, g_1, h, h_1, \ldots, h_n$ $\in G_1$ randomly, where $n$ is the number of attributes in universe. $H_1: \{0,1\}^* \rightarrow G_1$ and $H_2: G_2 \rightarrow \{0,1\}^*$ are two secure hash functions. TA publishes $PK = \{G_1,G_2,g,g_0,g_1,h,h_1,\ldots,h_n,H_1,H_2\}$ as system public parameter, $SK = \{\mathbb{E},A\}$ and keeps the master secret key $MSK = \{\mathbb{E},A\}$.

OABE $\textit{KeyGen}_{init}(A,MSK)$: Upon receiving a private key request on access policy $A$, TA selects $x_1 \equiv Z_p$ randomly and computes $x_2 \equiv g_1^x \mod p$. $OKGCSP \equiv (x_2)$ is sent to KG-CSP to generate outsourcing private key $SKGCSP \equiv OKTA \equiv x_2$ is used to generate local private key $SKTA$ at TA side.

OABE - $KeyGen_{out}(A,OKGCSP)$: TA sends $OKGCSP$ to KG-CSP for generating outsourcing private key $SKGCSP$. Upon receiving the request on $(A,OKGCSP)$, KG-CSP chooses a $d \equiv 1$ - degree polynomial $q(g)$ randomly such that $q(0) \equiv x_1$. For $\forall A, KG-CSP$ chooses $r \equiv Z_p$ randomly, and computes $d_0 \equiv g_{i1}^{q(g,h_1)}$ and $d_1 \equiv g^r$. KG-CSP sends outsourcing private key $SKGCSP = \{d_0,d_1\}$ to TA.

OABE - $KeyGen_{in}(OKTA)$: TA takes $OKTA$ as input and computes $d_{0} \equiv g^{r_2}(g,h)^r_1$ and $d_{1} \equiv g^r$, where $r_1 \equiv Z_p$ is selected randomly, $r_0$ is the default attribute. TA sets private key $SK = (SKGCSP,SKTA)$, where $SKTA = \{d_0,d_1\}$. TA responds the user with $SK$ by secure channel.

KSF - $KeyGen(PK,MSK,A,qBF)$: To get a query private key of DU with access policy $A$, DU and TA interacts as follows:

1. DU chooses a blinding factor $BF \equiv \{0,1\}^*$ randomly, and provides a commitment $qBF \equiv g^{qBF}_1$ to an access policy $A$. DU keeps $u$ secret.

2. TA retrieves $(g,h)^\mathbb{E}_A$ corresponding to $A$, and computes a query private key $QK = g^{qBF}_1 \equiv (g,h)^\mathbb{E}_A$ for DU.

3. TA sends the query private key $QK$ to DU by secure channel.

Encrypt(M,PK,SK): It takes as input a message $M$, $\mathbb{H}(\mathbb{E})$, the public parameters PK and an attribute set $\mathbb{E}^*$ associated with ciphertext. $\mathbb{E}$ randomly selects $s \equiv Z_p$ and computes $C_0 \equiv \mathbb{E}(g,g_2)^f$, $C_i \equiv g^s_i$, $C_{i1} \equiv g_h^s_i$ for each $i \in \mathbb{E}$, $C_{i0} \equiv \mathbb{E}(g_0,h)^s_i$. DU outputs the ciphertext with attribute set $\mathbb{E}^*$, where $CT = (\mathbb{H}U \equiv (\mathbb{H},C_i,C_0,C_{i0} \equiv C_{i1} \equiv C_0)$.

Index(PK,CT,KW): DU selects $r \equiv Z_p$ randomly and runs the index generation algorithm to compute $k_i = (e(g_2,g_1)^r_1 \cdot e(g,h_i)(kw)^r_2 \cdot C_i)^{1/m}$ for each $kw \equiv KW$ where $\{1,\ldots,m\}$. DU outputs the indexes of keywords set as $\{IX(KW) = (K_i,k_i,K_j)\}$ for $kw \equiv KW$ where $K_1,K_2,K_2 \equiv \{0,1\}^*$ and $K_1 \equiv H_1(k)$. DU upload 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 $T_{kw}(kw) \equiv H_1(kw)^QK$ and sets $I \equiv \{I_0, I_1, I_2 \equiv d_0, I_3 \equiv d_1\}$ for all $\forall A$, $D_i \equiv d_{1i}$, DU sets trapdoor for the keyword $kw$ as $T_{kw} \equiv T_{\mathbb{E}}(kw,I,D)$. DU requests $\text{Test}(IX(KW),T_{kw},CT)$: 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 in ciphertext satisfies the access policy $A$, D-CSP downloads all those ciphertext and executes partial decryption for them. DCSP computes:
D-CSP searches for the corresponding ciphertext CT related to the appointed index of keywords through submitted trapdoor kw. D-CSP computes:

\[ k_{iw} = \frac{e(K_i, T_q(kw))}{e(D_i, K_2)} = e(g_1, g_2)^y \cdot e(g, H_i(kw))^y, \]

and \( H_2(k_{iw}) \). D-CSP obtains the matching ciphertext by comparing \( H_2(k_{iw}) \) with each tuple \((CT, IX(KW))\) stored in S-CSP. D-CSP tests whether \( H_2(k_{iw}) \) exists. If not, 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_{SA}):** Upon receiving the \( Q_{CT} \) and the CT from D-CSP, DU can completely decrypt the ciphertext and obtain the message

\[ M = C_0 \cdot e(d_{\theta_1}, C_0) \cdot Q_{CT} \cdot e(C_1, d_{\theta_0}). \]

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

**TABLE 2 Size of each Value**

<table>
<thead>
<tr>
<th>PK</th>
<th>MK</th>
<th>SK</th>
<th>TK</th>
<th>RK</th>
<th>CT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCL 13 [4]</td>
<td>( n \times 4 )</td>
<td>( G_1 )</td>
<td>( Z_P )</td>
<td>( 2N_2 \times G_1 )</td>
<td>( (N_1 + 2) \times G_1 )</td>
</tr>
<tr>
<td>LHL 14 [3]</td>
<td>( n \times 4 )</td>
<td>( G_1 )</td>
<td>( Z_P )</td>
<td>( 2N_2 \times G_1 )</td>
<td>( (N_1 + 2) \times G_1 )</td>
</tr>
<tr>
<td>GHW 13 [8]</td>
<td>( 4 \times G_1 )</td>
<td>( Z_P )</td>
<td>none</td>
<td>( N_1 + 1 \times G_1 )</td>
<td>( 2N_2 \times G_1 )</td>
</tr>
<tr>
<td>Our scheme</td>
<td>( n \times 4 )</td>
<td>( G_1 )</td>
<td>( Z_P )</td>
<td>( 2N_2 \times G_1 )</td>
<td>( (N_1 + 4) \times G_1 )</td>
</tr>
</tbody>
</table>

**TABLE 3 Computational cost**

<table>
<thead>
<tr>
<th>Encrypt</th>
<th>Transform</th>
<th>Decrypt</th>
<th>DecryptOut</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCL 13 [4]</td>
<td>( C_0 + 2G_1 + (3 + 2N_1)G_1 )</td>
<td>none</td>
<td>( 2N_1C_0 + (2N_1 + 1)G_1 )</td>
</tr>
<tr>
<td>LHL 14 [3]</td>
<td>( C_0 + 2G_1 + (3 + 2N_1)G_1 )</td>
<td>none</td>
<td>( 2N_1C_0 + (2N_1 + 3)G_1 )</td>
</tr>
<tr>
<td>GHW 13 [8]</td>
<td>( C_0 + (N_1 + 1)G_1 + 3G_1 + N_1H )</td>
<td>( 2N_2G_1 )</td>
<td>( 2(2N_1 + 1)C_0 + (2N_1 + 3)G_1 )</td>
</tr>
<tr>
<td>Our scheme</td>
<td>((1 + K)C_0 + (1 + K)G_1 + (3 + 2N_1)G_1 + KH)</td>
<td>( 4G_1 )</td>
<td>( 2(N_1 + 1)C_0 + (C_0 + G_1) )</td>
</tr>
</tbody>
</table>

**5. PERFORMANCE ANALYSIS**

**5.1 Complexity Analysis**

In Table 2 and Table 3, we briefly compare our scheme with the others.

**5.2 Efficiency Analysis**

We compared the performance of the four stages in our experiment. We simulated with the java pairing-based cryptography (JFBC) library version 2.0.0, which is a port of the pairing-based cryptography (PBC) library 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 \( l \times d \geq 1024 \). We implement our scheme on Type A curve \( y^2 = \frac{x^1}{x}, \) where \( p \) is 160 bits, \( l = 512 \). We select SHA- as the hash function. We implement our scheme and the scheme [4] 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 [4]. 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 [4] 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 [4], but the computational cost in our system keeps in a low level.

6. CONCLUSION

In this article, we propose the function of Integrity Verification (Verifiability) which can greatly protect the security and privacy of users by checking the data received by them was the actual data uploaded by data owner. 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. Actually, we are easy to extend our KSF-OABE scheme to support access structure represented by tree.

7. FUTURE WORK

Keyword Search Function (KSF) is an important feature of OABE which allows the data user to search for required data in the vast data, so one of our future works is to construct OABE which can provide KSF and make it KSF-OABE. 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

I am extremely thankful to Mr. M. Nagaraju, Associate Professor, Project Coordinator and Internal Guide, Department of CSE, for his constant guidance, encouragement and moral support throughout the project. Without his support, I can't do this project.

I express my thanks to all staff members for all the help and co-ordination extended in bringing out this Project. Finally, I am very much thankful to my family who guided me for every step.

8. REFERENCES


