International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

INTEGRITY VERIFICATION AND ATTRIBUTE BASED ENCRYPTION FOR CLOUD STORAGE

V. Venkata Surya Deep[1], M. Nagaraju[2]

[1]Student, Dept. of Computer Science Engineering, BENAIAH Institute of Technology and Science, Burugupudi, Rajamahendravaram, Andhra Pradesh, India.
[2]Associate Professor, Dept. of Computer Science Engineering, BENAIAH 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, Qianetal. 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. Qianetal. 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, w e organically integrates 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 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

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 *G1* and *G2 be* multiplicative cyclic groups with prime order *p.* Suppose *g* is a generator of *G*1. *e*: *G*1 *G*1*G*2 is a bilinear map if it satisfies the following properties:
(1) Bilinearity: For all *u*,*v**G1*, $e\,(u^{a}, u^{b}) = e\,(u,v)^{ab}$ where *a*,*b* $_p$ are selected randomly.
(2) Nondegeneracy: There exists *u*,*v* G_1 such that *e*(*u*,*v*) 1 .
(3) Computability: For all *u*,*v*G_1 , there is an efficient algorithm to compute *e*(*u*,*v*).

2.3 Complexity Assumption

Definition1(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*)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

Definition2 (Access Structure).

Suppose { $P_1,\ldots,P_n$ } are a set of parties. A collection *A*$2^{\{P1,..,Pn\}}$is monotone if *B*,*C* , *B**A* and *B* *C* then *C**A* . 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 and *A* be an attribute set and access policy. A predicate (, *A*) is defined as follows:

□□(□,*A*)□{0,1} , if □□□*A* , the value of □□(□, *A*) equals to 1 , else the value is 0 .

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.

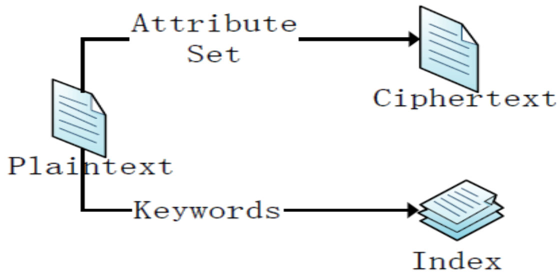
Key Generation Cloud Service Provider (KG-CSP). It is a participant that supplies outsourcing computing service for TA by completing the costly key generation tasks allocated by TA.

Decryption-Cloud Service Provider (D-CSP). It is a participant that supplies outsourcing computing service through accomplishing partial decryption for 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 □□(□,*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_{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 □□. We have (I_{enc}, I_{key}) = (□□*A*) where □□and *A* are attribute set and access structure respectively. We describe the formal definition of OABE scheme as follow:

***Setup*(□)** : TA runs the *Setup* algorithm, which takes a security parameter □□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□) □□(*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*) □□(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}. KGCSP returns TA the SK_{KGCSP}. It is described as *OABE* □□$KeyGen_{out}$ (*A*, OK_{KGCSP}) □□ 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})□□□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}) □□*QK.*

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

***Index* (*PK, CT, KW*)**: This algorithm is performed by DO which takes the system public parameter *PK,* and a keyword set *KW* kw$_i$}$_{i-1}$ 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 T_{kw} corresponding to the keyword *kw*. It is described as *Trapdoor (PK, QK, BF, kw)* T_{kw}.

***Test (IX (KW), Tkw, 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 as input. If the satisfies the access policy *A* embedded in *CT.* D-CSP partially decrypts the *CT* to get Q_{CT}. D-CSP searches for the corresponding ciphertext *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 *Test (IX (KW),* T_{kw}*, CT)*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}*)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 SK_{KGCSP}, 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 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 SCSP. 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.

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* - $KeyGen_{out}$ 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 *KGCSP SK* 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 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 .

Challenge: The adversary sends two messages M_0, M_1 with equal-length and a challenge attribute set *to the challenger, subject to the restriction that, *can not satisfy *A A*. The challenger chooses {0,1} , and runs

Encrypt(*PK*,$M_{□}$,□□*) □□*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 □*.

(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 □'□□{0,1} for □□.The advantage which the adversary can win the game is defined as | *Pr* (□'□= □) - 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* (□'□= □) - 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 □*□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 an tree-based access policy bound up with user private key, □□is an attribute set embedded in ciphertext, *U* is the attribute universe, and *d* is a threshold value set in advance. If □(□,*A*) □1, *S* is a attribute set which satisfies *S* □{□□*A*}□| *S* |□□*d* .

Setup (□): 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 □□G_2 □□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* □ Z_p, computes g_1 □□g^x, and chooses g_2, *h*, *h1*,.., *hn* □□G_1 randomly where *n* is the number of attributes in universe. H_1 :{0,1}* □G_1 and H_2: G_2 □$\{0,1\}^{\log p}$ □are two secure hash functions. TA publishes *PK* □□□ G_1, G_2, g, g1, g2, h, h1,...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 □□Z_p randomly and computes x_2 □□*x* □□x_1 mod *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 KGCSP for generating outsourcing private key SK_{KGCSP}. Upon receiving the request on (*A*, OK_{KGCSP}), KG-CSP chooses a (*d* □1) - degree polynomial *q*(□) randomly such that *q*(0) □□x_1 . For *i*□*A*, KG-CSP chooses r_i □□Z_p randomly, and computes d_{i0} = $g_2^{q(i)}(g_1h_1)^{r1}$ and d_{i1} = g^{r1}. KG-CSP sends outsourcing private key SK_{KGCSP} = $\{d_{i0},d_{i1}\}_{i□□}$ to TA.

OABE - KeyGen$_{in}$ (*OK_{TA}*): TA takes OK_{TA} as input and computes $d_{□0}$ □ g^x2 $(g_1h)^{re}$ □and $d_{□1}$ = g^{re} , where $r_{□}$ □□ Z_p□□□is selected randomly, □□is the default attribute. TA sets private key *SK* = (SK_{KGCSP}, SK_{TA}), where SK_{TA} = {$d_{□0}$, $d_{□1}$}. TA responses 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 follow:

— DU chooses a blinding factor *BF* □□*u*□Z^*_p randomly, and provides a commitment q_{BF} □□$g^{1/2}{}_u$ and an access policy *A* to TA. DU keeps *u* secret.

— TA retrieves $(g_1h)^re$ corresponding to *A,* and computes a query private key QK = $g^{x/u}{}_2$ $(g_1h)^re$ for the DU.

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

Encrypt(*M*,*PK*,□) : It takes as input a message *M* □□G_2 , the public parameters PK and an attribute set □□□associated with ciphertext. DO randomly selects *s* □□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*□□□, $C_{□}$ □□□$g_1h)^s$. DO outputs the ciphertext with attribute set □□, where *CT* = (□□U {□}, C_0□ C_1□□ C_i□$_{i□□}$,□ $C_{□}$*).*

Index(*PK*,*CT*,*KW*) : DO selects *r* □□Z_p randomly and runs the index generation algorithm to compute k_i = $e(g_1,g_2)^s$ .e(g,H_1(kw_i))^s □□G_2 for each kw_i □*KW* where *i* □1,...,*m*. DO outputs the indexes of keywords set as *IX*(*KW*)=(K_1,K_2,K_i) for kw_i □□*KW* where K_1 □C_1 □□g^s, *K*2 □□ $C_{□}$□□□□g_1h□^{s}□ , K_i □□H_2 (k_i). DO 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 $T_q(kw)$ □□$H_1(kw)QK^u$,and sets *I* □ (I_{i0} □□d_{i0}, I_{i1} □□d_{i1}) for all *i*□*A* , D_1 □ $d_{□1}{}^u$. 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 ciphertext and executes partial decryption for them. DCSP 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 *kw T*. 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 □□*KW* .D-CSP outputs □□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.

$$Q_{CT} = \frac{\prod_{i\in S} e(C_1, I_{i0})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(I_{i1}, C_i)^{\Delta_{i,S}(0)}} = \frac{\prod_{i\in S} e(g^s, g_2^{q(i)}(g_1 h_i)^{r_i})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(g^{r_i}, (g_1 h_i)^s)^{\Delta_{i,S}(0)}}$$

$$= \frac{e(g, g_2)^{s\sum_{i\in S} q(i)\Delta_{i,S}(0)} \prod_{i\in S} e(g^s, (g_1 h_i)^{r_i})^{\Delta_{i,S}(0)}}{\prod_{i\in S} e(g^{r_i}, (g_1 h_i)^s)^{\Delta_{i,S}(0)}}$$

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

$$k_{kw} = \frac{e(K_1, T_q(kw))}{e(D_1, K_2)} = \frac{e(g^s, H_1(kw)QK^u)}{e(d_{\theta 1}{}^u, (g_1 h)^s)}$$

$$= \frac{e(g^s, H_1(kw) g_2^x (g_1 h)^{r_\theta u})}{e(g^{r_\theta u}, (g_1 h)^s)}$$

$$= \frac{e(g^s, (g_1 h)^{r_\theta u}) e(g, H_1(kw))^s e(g, g_2)^{sx}}{e(g^{r_\theta u}, (g_1 h)^s)}$$

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

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

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

$$= M$$

5. PERFORMANCE ANALYSIS

5.1 Complexity Analysis

In Table2 and Table3, we briefly compare our scheme

TABLE 2 Size of each Value

	PK	MK	SK		TK	RK	CT
LCL 13 [4]	(n □□4) $\lvert G_1 \rvert$	$\lvert Z_P \rvert$	$2N_2 \lvert G_1 \rvert$	$(N_1+2) \lvert G_1 \rvert + \lvert G_T \rvert$	$(2N_2 - 1)\lvert G_1 \rvert$	$2 \lvert G_1 \rvert$	$2 \lvert G_1 \rvert + 2 \lvert G_T \rvert$
LHL 14 [3]	(n □□4) $\lvert G_1 \rvert$	$\lvert Z_P \rvert$	$2N_2 \lvert G_1 \rvert$ □□$\lvert$□$Z_P \rvert$	$(N_1+2) \lvert G_1 \rvert + \lvert G_T \rvert$	$2N_2 \lvert G_1 \rvert$	$\lvert Z_P \rvert$	$\lvert G_T \rvert$
GHW 13[8]	$4 \lvert G_1 \rvert$	$\lvert Z_P \rvert$	none	$(N_1+1) \lvert G_1 \rvert + \lvert G_T \rvert + n \lvert Z_P \rvert$	$2N_2 \lvert G_1 \rvert$	$\lvert Z_P \rvert$	$2 \lvert G_T \rvert$
Our scheme	(n □□4) $\lvert G_1 \rvert$	$\lvert Z_P \rvert$	$2N_2 \lvert G_1 \rvert$	$(N_1+4) \lvert G_1 \rvert + (1+K) \lvert G_T \rvert$	$2N_2 \lvert G_1 \rvert$	$\lvert Z_P \rvert$	$2 \lvert G_T \rvert$

TABLE 3 Computational cost

	Encrypt	*Transform*	*Decrypt$_{Out}$*	*Decrypt*
LCL 13 [4]	$C_e + 2G_T + (3+2N_1)G_1$	none	$2N_1C_e + (2N_1+1) G_T$	$2C_e + 3G_T$
LHL 14 [3]	$C_e + 2G_T + (3+2N_1)G_1$	$2N_2 G_1$	$2(N_1+1)C_e + (2N_1+3) G_T$	$3G_T$
GHW 13[8]	$C_e + (N_1+1)G_1 + 3G_T + N_1H$	$2N_2 G_1$	$(N_1+1)C_e + 2N_1G_1 + N_1G_T$	$2G_T$
Our scheme	$(1+K)C_e + 2(1+K)G_T + (3+2N_1)G_1 + KH$	$4G_1$	$2(N_1+1) + (C_e + G_T)$	$2C_e + 3G_T$

5.2 Efficiency Analysis

We compared the performance of the four stages in our experiment is simulated with the java pairing-based cryptography (JPBC) 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* □□x^2 □□x^3 , 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

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

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

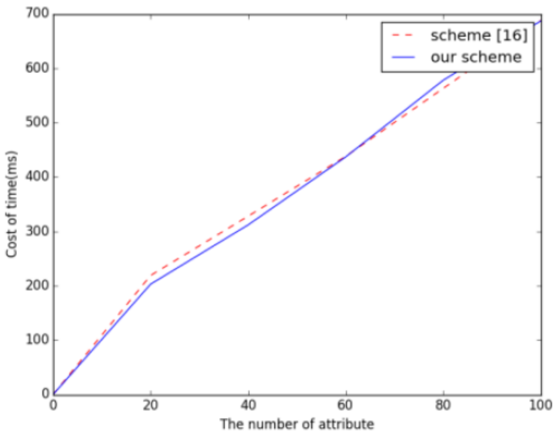


Fig.2(a) Setup

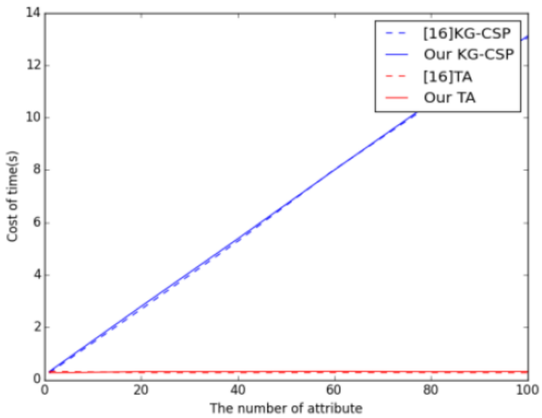


Fig.2 (b) KeyGen

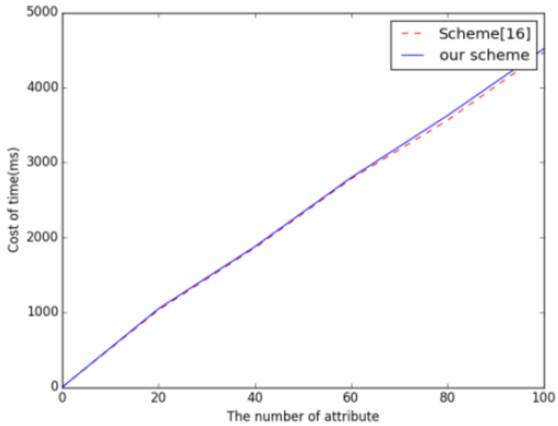


Fig.2(c) Encryption

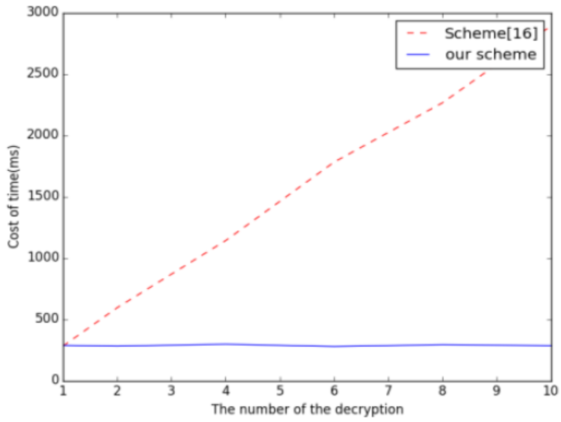


Fig.2(d) Decryption

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 done 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

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] Jiguo Li, Xiaonan Lin, Yichen Zhang and Jinguang Han, "KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage," *IEEE Transactions on Services Computing* (Volume: 10, Issue: 5, Sept.-Oct. 1 2017).

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