ENSURING SCALABLE, SECURED, MAINTENANCE AND ACCESS CONTROL OF COLLEGE DATA, USING CLOUD COMPUTING

Yogeesh A.C[1], Dilip B.R[2], Chandra [3], Divya S Abhyankar[4]

[1] Assistant Professor & [2, 3&4] Under Graduate Student,
Department of Computer Science and Engg., Government Engg. College Kushalnagar, India

ABSTRACT

Cloud computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. With the rapid development of information and communication technique, there is a growing interest in harnessing the inexpensive educational data in colleges. It is changing the way; colleges manage their data, due to its robustness, low cost and ubiquitous nature. By data outsourcing, colleges can be relieved from the burden of local data storage and maintenance. However, the fact those colleges no longer have physical possession of the possible large size of outsourced data integrity protection in cloud computing a very challenging and potentially formidable task especially for users with constrained computing resources and capabilities. Privacy security is a key issue of cloud storage. Encryption is a well-established technology for protecting sensitive data. This paper addresses how to ensuring scalable, secured, maintenance and access control of college data storage in Cloud Computing. The problem of simultaneously achieving fine-grainedness, scalability, and data confidentiality of access control still remains unresolved. To overcome this problem, we defining and enforcing policies based on data attributes and allowing data owner to delegate most of the computation tasks involved in fine-grained data access control to untrusted cloud servers without disclosing the underlying data contents. We achieve this goal by exploiting and uniquely combining techniques of attribute-based encryption (ABE), proxy re-encryption and lazy re-encryption.

KEYWORDS: *Cloud Computing, Encryption, Attribute-based encryption (ABE), Proxy re-encryption, Lazy re-encryption.*

I. INTRODUCTION

Cloud computing is a promising computing paradigm which recently has drawn extensive attention from both academic and industry. In which resources in the computing infrastructure are provided as services over the Internet. Along with this new paradigm, various business models are developed, which can be described by terminology of "X as a service (XaaS)" [1] where X could be software, hardware, data storage, and etc. Cloud storage provides on-demand, scalable and QoS guaranteed storage resource, and users can operate their data anytime and anywhere. Successful examples for the cloud storage providers are Amazon's EC2 and S3 [2], Google App Engine [3], and Microsoft Azure [4] which provideusers with scalable resources in the pay-as-you use fashion at relatively low prices. For example, Amazon's S3 data storage service just charges \$0.12 to \$0.15 per gigabyte-month. As compared to building their own infrastructures, users are able to save their investments significantly by migrating businesses into the cloud. With the increasing development of cloud computing technologies, it is not hard to imagine that in the near future more and more businesses will be moved into the cloud.In this paper, we address the open issue and purpose of Scalability, Secured, Maintenance and Access Control of College related Data scheme in Cloud Computing. Data considered with college is considered as very precious and very important. When we consider college

data its storage and maintenance is considered to be very risky job. The traditional way of storing the data and its maintenance is a tedious job. College data can be considered as such data, assignments, exam model papers, results, notes, meeting information, student information etc… Here we are using main Hybrid cloud to store a data, which consists of both Public cloud and Private cloud. An example is using commodity resources from a public cloud such as web servers to display non sensitive data, which interact withSensitive data stored or processed in a private cloud. Public cloud involves an organization using a vender's cloud infrastructure which is shared via the internet with many other organization and other members of the public. This model has maximum potential cost efficiencies due to economies of scale. Data that are considered as public, such as student information, academic results, overall college progress, college history, infrastructure details, courses offered etc..this will be stored in public cloud. Private cloud involves an organization's exclusive use of cloud infrastructure and services located at the organization's premises or offsite, and manage by the organization or a vender. Compared to public cloud model, the private cloud model has reduced potential cost efficiencies [5]. Data that are considered as confidential or private such as, employee's details, meeting information, employee's salary information, etc…are stored in private cloud. Data that are stored in private cloud can be accessed by only privileged users, the privilege are provided by setting attribute to the file, by the owner of the data who outsourced that data to be on cloud. The access structure of each user can thus be defined as a unique logical expression over these attribute to reflect the scope of data files that the user is allowed to access. For the purpose of helping the data owner enjoy fine-grained access control of data stored on untrusted cloud servers, a feasible solution would be encrypting data through certain cryptographic primitive(s), and disclosing decryption keys only to authorized users. Unauthorized users, including cloud servers, are not able to decrypt since they do not have the data decryption keys. This general method actually has been widely adopted by existing works [6]–[7] which aim at securing data storage on untrusted servers. The data owner who outsources the data should decide whether the data should be in private cloud or public cloud.

Main contributions of this paper can be summarized as follows:-

1) This paper simultaneously achieves scalability, secured, maintenance and data confidentiality for data access control of college related data in cloud computing;

2) Our proposed scheme enables the data owner to delegate most of computation intensive tasks to cloud servers without disclosing data contents or user access privilege information;

3) The proposed scheme is provably secure under the standard security model. In addition, our proposed scheme is able to support user accountability with minor extension.

ORGANIZATION OF REPORT:

Section one gives brief introduction about Cloud Computing and gives pictures of different Clouds that are used in Cloud Computing environments and explains basic Idea behind this work. It gives the overview of the data owners, clients who are in interest to use this data. It also gives the overview of the clouds that are used to store the data depending on their privacy level.

Processing stages to store college data in private cloud; this section enlightens the three basic process such as data outsourcing, data storage and data retrieving. Data outsourcing section gives the basic idea of how the data owner will outsource the data by setting the attributeusing key policy attribute-based encryption (KP-ABE), to the data so that only privileged users can access the data .Data Storage, Cloud Servers keeps an attribute history list AHL which records the version evolution history of each attribute and Proxy Re-Encryption (PRE) keys used and user list UL which records IDs of all the valid users in the system. Data Retrieving, Cloud Servers respond user request for data access, and update user secret keys and re-encrypt requested data files if necessary.

Technique preliminaries section gives the overview of basic techniques used in order to achieve secure, scalable and fine-grained access control on outsourced data in the cloud that combines the following three advanced cryptographic techniques: KP-ABE, PRE and lazy re-encryption.

Basic operation and required algorithm this section enlightens the basic file operations such as, System Setup, New File Creation, New User Grant, and UserRevocation, File Access, File Deletion, and the interaction between involved parties and the algorithms required for the basic operation i.e., ASetup, AEncrypt, AKeyGen, ADecrypt, AUpdateAtt, AUpdateSK, AUpdateAtt4File, and AMinimalSet.

II. BASIC ARCHITECTURE

The basic architecture is composed of the following parties: Hybrid Cloud, Clients, many Cloud Servers, and a Third Party Auditor if necessary [1]. To access data files shared by the data owner, Data Consumers download data files of their interest from cloud Servers and decrypt the data using secret key. Neither data owner nor users will be always online. They come online just on the necessity basis.Cloud storage servers are always online and operated by Cloud Service Provider (CSP).They are assumed to have abundant storage capacity and computation power. The Third partyAuditor is also `an online party which is used for auditing every file access event [8].
A representative basic architecture for cloud data storage is illustrated in Fig. 1. The different entities in basic architecture can be identified as follows:

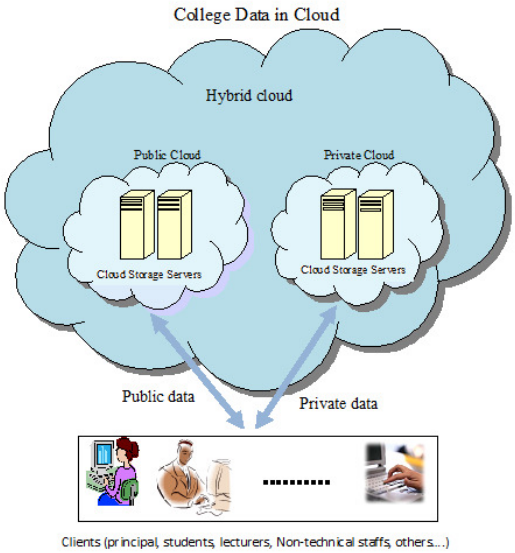

Fig. 1: Cloud data storage architecture

Client:an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations [9]. In our proposed scheme, Clients may be a Data Owner or Data Consumers. They are illustrated as follows:

- ***Data Owner:***Data Owner isthe person who authorized of the data so that he stores the data in the Cloud. The Data owner can be lecturers, principal, students, non-technical staff etc…The data that the data owner wants to be in cloud can be of two specific characters such as public and private, and depending on the data that they want to store they can store their data in private cloud or public cloud[7].
- ***Data Consumers:*** Data Consumers are the person who is in need of the data i.e. stored in the cloud. They are provided with login id and a password so that they can login effectively. If they want the private data to be viewed, they should have the privilege. The privilege is set by the data owner during data uploading using file attribute method. The data consumer can the students, lecturers, others etc…they are also allowed to view data i.e. stored in the public cloud[6].

Hybrid cloud: includes both public and private clouds which stores college data that can be accessed in an effective way. That are illustrate as follows.

- ***Public Cloud:*** it stores the data that can be accessed by any clients. In general the client can be students, public peoples, parents etc…they are authorized to access the college related data such as infrastructure information of college, college progress etc…
- ***Private Cloud:*** it stores the data that can access only by authorized clients. The authorization to the client is set by the Data owner by setting file attribution to the data that they store in the Private Cloud. The data that are stored in private cloud can be lecturer's notes, information, meeting information, student opinions etc… The clients are allowed to view the data using their login id and respective passwords. To decrypt the data that they want is done through the

decryption key. For ex, let us consider the meeting information, this information be authorized by the principal of the college so that he wants only the HOD's of respective branch wants to attended the meeting. The meeting information may contain information such as meeting date, place, timing, agenda etc. The data owner, i.e. the principal will set the attribute to the data that he stores so that only the authorized person's such as HOD's of Computer Science Department, Mechanical department are able a view the data. Suppose an unauthorized person such as students, lecturer's attempts to view this data they are not allowed to view because the data owner as not set any attributes to this unauthorized viewers. They a detected as fraud or unauthorized users and their information will be available in the fraud user list.

Cloud Storage Server (CSS):an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the clients' data [9].

Third Party Auditor (TPA): an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request [9].

III. PROCESSING STAGES TO STORE COLLEGE DATA IN PRIVATE CLOUD

Stage 1: Data outsourcing

In this stage, Data owner set the user access structure and encryption of data file is performed.Before this process, the Data owner authorization step is involved. If he is a new user, then he should create account with name and password and also mention their user category (For ex: Principal, HODs, Lecturer, etc). In the server side, it checks the Data owner name and its password for security purpose. If it is satisfied, for each data file the owner assigns a set of meaningful attributes which are necessary for access control. Different data files can have a subset of attributes in common. Each attribute is associated with a version number for the purpose of attribute update. In addition to these meaningful attributes, we also define one dummy attribute, denoted by symbol *AttD* for the purpose of key management. *AttD* is required to be included in every data file's attribute set and will never be updated. The access structure of each user is implemented by an access tree. Interior nodes of the access tree are threshold gates. Leafnodes of the access tree are associated with data file attributes. For the purpose of key management, we require the root node to be an *AND* gate (i.e., *n*-of-*n* threshold gate) with one child being the leaf node which is associated with the dummy attribute, and the other child node being any threshold gate. The dummy attribute will not be attached to any other node in the access tree. Using key policy attribute-based encryption (KP-ABE), data are associated with attributes for each of which a public key component is defined [8]. The encryptor associates the set of attributes to the message by encrypting it with the corresponding public key components.Fig.2 gives the detailed overview of private cloud and User access structure.

Stage 2: Data storage

Cloud Servers stores data and all the secret key components of *SK* except for the one corresponding to the dummy attribute *AttD*. Such a design allows Cloud Servers to update these secret key components during user revocation. If only there is a file data access request from a user, do Cloud Servers re-encrypt the requested files and update the requesting user's secret key. This statistically saves a lot of computation overhead since Cloud Servers are able to "aggregate" multiple update/re-encryption operations into one if there is no access request occurring across multiple successive user revocation events. Cloud Servers also keeps an attribute history list *AHL* which records the version evolution history of each attribute and *Proxy Re-Encryption (PRE)* keys used and user list *UL* which records *ID*s of all the valid users in the system [8].

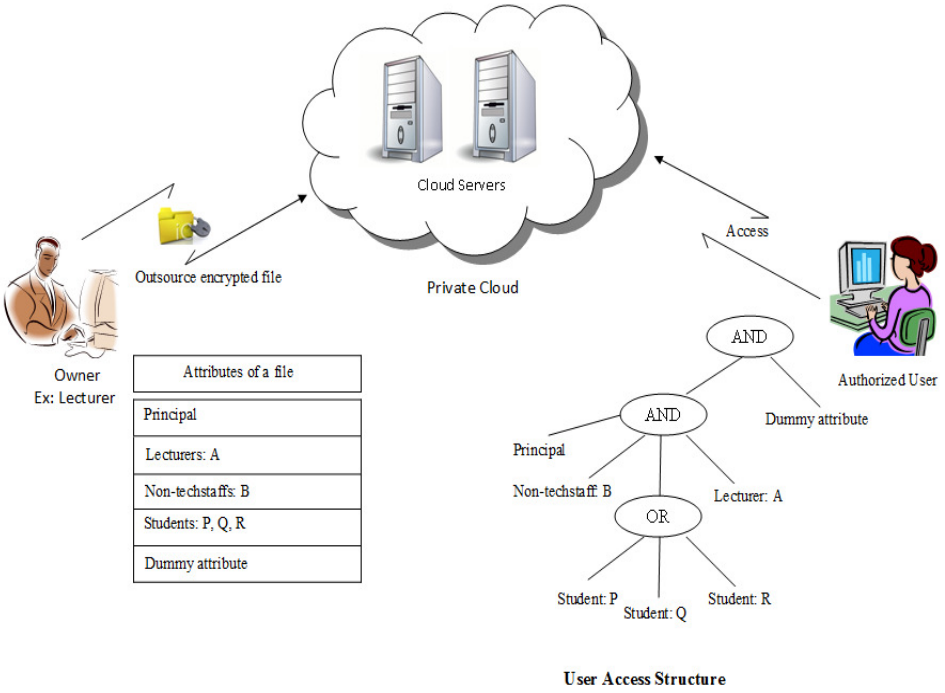

Figure 2: The above figure shows the detailed overview of private cloud.

Stage 3: Data retrieving

Cloud Servers respond user request for data access, and update user secret keys and re-encrypt requested data files if necessary. As is depicted in Fig. 2, Cloud Servers first verify if the requesting useris a valid system user in system user list. Cloud Servers will not perform update/re-encryption if secret key components/data files are already of the latest version. Finally, Cloud Servers send updated secret key components as well as cipher texts of the requested data files to the user. On receiving the response from Cloud Servers, the user first verifies if the claimed version of each attribute is really newer than the current version he knows. For this purpose, he needs to verify the data owner's signatures on the attribute information (including the version information) and the corresponding public key components. If user unable to verify with the attribute of file or if the private that has been generated is found to wrong, then user will be displayed with error message ("Error Message: Your are not an authenticate user").

IV. TECHNIQUE PRELIMINARIES

In order to achieve secure, scalable and fine-grained access control on outsourced data in the cloud, we utilize and uniquely combine the following three advanced cryptographic techniques: KP-ABE, PRE and lazy re-encryption. More specifically, we associate each data file with a set of attributes, and assign each user an expressive access structure which is defined over these attributes. To enforce this kind of access control, we utilize KP-ABE to escort data encryption keys of data files. Such a construction enables us to immediately enjoy fine-grainedness of access control.

A. Key Policy Attribute-Based Encryption (KP-ABE):

KP-ABE [10] is a public key cryptography primitive for one-to-many communications. In KP-ABE, data are associated with attributes for each of which a public key component is defined. The encryptor associates the set of attributes to the message by encrypting it with the corresponding public key components. Each user is assigned an access structure which is usually defined as an access tree over data attributes, i.e., interior nodes of the access tree are threshold gates and leaf nodes are associated with attributes. User secret key is defined to reflect the access structure so that the user is able to

decrypt a cipher text if and only if the data attributes satisfy his access structure. A KP-ABE scheme is composed of four algorithms which can be defined as follows:

Setup This algorithm takes as input a security parameter κ and the attribute universe $U = \{1, 2, \ldots, N\}$ of cardinality N. It defines a bilinear group $\mathbb{G}_1$ of prime order p with a generator g, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ which has the properties of bi-linearity, computability, and non- degeneracy.It returns the public key PK as well as a system master key MK as follows

$PK = (Y, T_1, T_2, \ldots, T_N)$

$MK = (y, t_1, t_2, \ldots, t_N)$

Where $T_i \in \mathbb{G}_1$ and $t_i \in \mathbb{Z}_p$ are for attribute i, $1 \leq i \leq N$, and $Y \in \mathbb{G}_2$ is another public key component. We have $T_i = g^{ti}$ and $Y = e(g, g)^y$, $y \in \mathbb{Z}_p$. While PK is publicly known to all
the parties in the system, MK is kept as a secret by the authority party.

Encryption This algorithm takes a message M, the public key PK, and a set of attributes I as input. It outputs the cipher text E with the following format:

$E = (I, \tilde{E}, \{E_i\}_{i \in I})$

Where $\tilde{E} = MY^s$, $E_i = T_i^s$, and s is randomly chosen from Z_p.

Key Generation This algorithm takes as input an access tree T, the master key MK, and the public key PK. It outputs a user secret key SK as follows. First, it defines a random polynomial $p_i(x)$ for each node i of T in the top-down manner starting from the root node r. For each non-root node j, $p_j(0) = p_{parent(j)}(idx(j))$ where $parent(j)$ represents j's parent and $idx(j)$ is j's unique index given by its parent. For the root node r, $p_r(0) = y$.

Then it outputs SK as follows.

$SK = \{sk_i\}_{i \in L}$

Where L denotes the set of attributes attached to the leaf nodes of T and $sk_i = g^{(pi(0)/ti)}$.

Decryption This algorithm takes as input the cipher text E encrypted under the attribute set I, the user's secret key SK for access tree T, and the public key PK. It first computes $e(E_i, sk_i) = e(g, g)^{pi(0)s}$ for leaf nodes. Then, aggregates these pairing results in the bottom-up manner using the polynomial interpolation technique. Finally, it may recover the blind factor $Y^s = e(g, g)^{ys}$ and output the message M if and only if I satisfies T.Please refer to [10] for more details on KP-ABE algorithms. [11] is an enhanced KP-ABE scheme which supports user secret key accountability.

B. Proxy Re-Encryption (PRE):

Proxy Re-Encryption (PRE) is a cryptographic primitive in which a semi-trusted proxy is able to convert a cipher text encrypted under Alice's public key into another cipher text that can be opened by Bob's private key without seeing the underlying plaintext. More formally, a PRE scheme allows the proxy, given the proxy re-encryption key $rk_{a \leftrightarrow b}$, to translate cipher texts under public key pk_a into cipher texts under public key pk_b and vice versa. Please refer to [12] for more details on proxy re-encryption schemes.

C. Lazy re-encryption:

The lazy re-encryption [13] technique and allow Cloud Servers to aggregate computation tasks of multiple operations. The operations such as

- Update secret keys
- Update user attributes

V. BASIC OPERATION AND REQUIRED ALGORITHM

1) Basic file operation:

We describe the implementation of high level operations, i.e., *System Setup*, *New File Creation*, *New User Grant*, and *UserRevocation*, *File Access*, *File Deletion*, and the interaction between involved parties.Fig.3 gives the description of notation to be used in our basic operation [8].

Notation	Description
PK, MK	system public key and master key
T_i	public key component for attribute i
t_i	master key component for attribute i
SK	user secret key
Sk_i	user secret key component for attribute i
E_i	ciphertext component for attribute i
I	attribute set assigned to a data file
DEK	symmetric data encryption key of a data file
P	user access structure
LP	set of attributes attached to leaf nodes of P
Att_D	the dummy attribute
UL	the system user list
AHL_i	attribute history list for attribute i
$Rk_i \leftrightarrow i'$	proxy re-encryption key for attribute i from itcurrent version to the updated version i
δ_O, X	the data owner's signature on message X

Fig. 3: Notation used in basic operation description

System Setup In this operation, the data owner chooses a security parameter κ and calls the algorithm level interface *A Setup* (κ), which outputs the system public parameter *PK* and the system master key *MK*. The data owner then signseach component of *PK* and sends *PK* along with these signatures to Cloud Servers.

New File Creation Before uploading a file to Cloud Servers, the data owner processes the data file as follows.

- select a unique *ID* for this data file;
- randomly select a symmetric data encryption key $DEK \xleftarrow{R} K$, where *K* is the key space, and encrypt the data file using *DEK*;
- define a set of attribute *I* for the data file and encrypt *DEK* with *I* using KP-ABE, i.e., $(\tilde{E}, \{Ei\}i \in I) \leftarrow AEncrypt(I, DEK, PK)$.

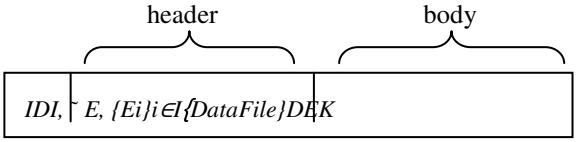

Fig. 4: Format of a data file stored on the cloud

Finally, each data file is stored on the cloud in the format as is shown in Fig.4.

New User Grant When a new user wants to join the system, the data owner assigns an access structure and the corresponding secret key to this user as follows [8].

- assign the new user a unique identity *w* and an access structure *P*;
- generate a secret key *SK* for *w*, i.e., $SK \leftarrow AKeyGen(P, MK)$;
- encrypt the tuple $(P, SK, PK, \delta_{O,(P,SK,PK)})$ with user *w*'s public key, denoting the ciphertext by *C*;
- send the tuple $(T, C, \delta_{O,(T,C)})$ to Cloud Servers, where *T* denotes the tuple $(w, \{j, sk_j\}_{j \in LP \setminus Att_D})$.

On receiving the tuple $(T, C, \delta_{O,(T,C)})$, Cloud Servers processes as follows.

- verify $\delta_{O,(T,C)}$ and proceed if correct;
- store *T* in the system user list *UL*;
- forward*C* to the user.

On receiving *C*, the user first decrypts it with his private key. Then he verifies the signature $\delta_{O,(P,SK,PK)}$. If correct,he accepts (P, SK, PK) as his access structure, secret key, and the system public key.

As described above, Cloud Servers store all the secret key components of *SK* except for the one corresponding to the dummy attribute *AttD*. Such a design allows Cloud Servers to update these secret key components during user revocation as we will describe soon. As there still exists one undisclosed

secret key component (the one for *AttD*),Cloud Servers cannot use these known ones to correctly decrypt ciphertexts. Actually, these disclosed secret key components, if given to any unauthorized user, do not give him any extra advantage in decryption as we will show in our security analysis.

User Revocation We start with the intuition of the userrevocation operation as follows. Whenever there is a user tobe revoked, the data owner first determines a minimal set ofattributes without which the leaving user's access structurewill never be satisfied. Next, he updates these attributes byredefining their corresponding system masterkey componentsin *MK*. Public key components of all these updated attributesin *PK* are redefined accordingly. Then, he updates user secretkeys accordingly for all the users expect for the one to be revoked. Finally, *DEK's* of affected data files are re-encrypted with the latest version of PK. The main issue with thisintuitive scheme is that it would introduce a heavy computationoverhead for the data owner to re-encrypt data files andmight require the data owner to be always online to providesecret key update service for users. To resolve this issue, wecombine the technique of proxy re-encryption with KP-ABEand delegate tasks of data file re-encryption and user secretkey update to Cloud Servers. More specifically, we divide theuser revocation scheme into two stages as is shown in Fig.5.

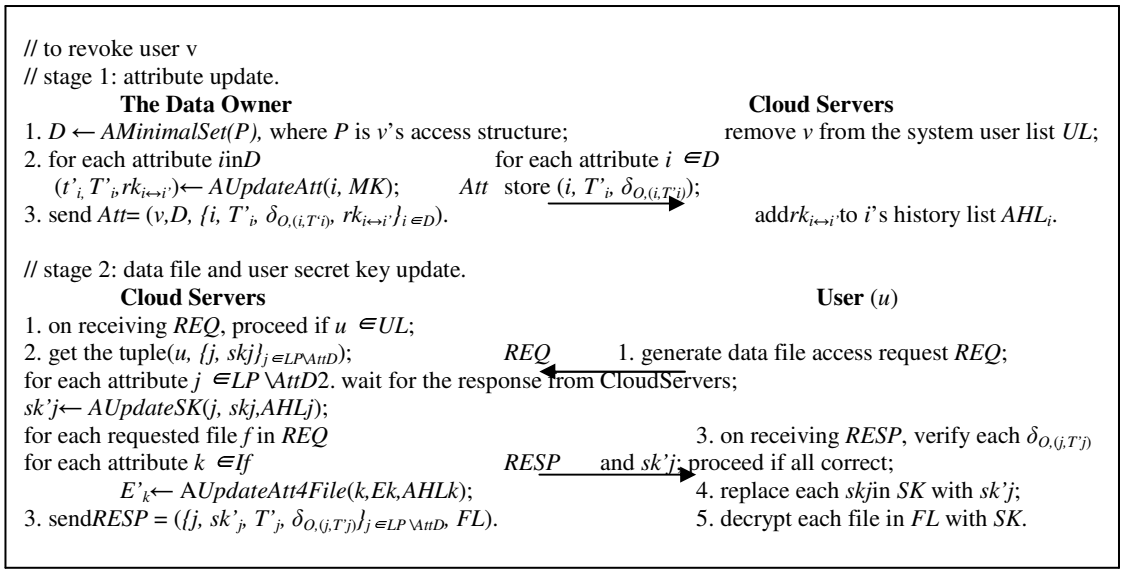

Fig. 5: Description of the process of user revocation.

In the first stage, the data owner determines the minimal setof attributes, redefines MK and PK for involved attributes,and generates the corresponding PRE keys. He then sendsthe user's ID, the minimal attribute set, the PRE keys, theupdated public key components, along with his signatures onthese components to Cloud Servers, and can go off-line again.Cloud Servers, on receiving this message from the data owner,remove the revoked user from the system user list UL, storethe updated public key components as well as the owner'ssignatures on them, and record the PRE key of the latestversion in the attribute history list AHL for each updatedattribute. AHL of each attribute is a list used to record theversion evolution history of this attribute as well as the PREkeys used. Every attribute has its own AHL. With AHL,Cloud Servers are able to compute a single PRE key thatenables them to update the attribute from any historical versionto the latest version. This property allows Cloud Servers toupdate user secret keys and data files in the "lazy" way asfollows. Once a user revocation event occurs, Cloud Serversjust record information submitted by the data owner as ispreviously discussed. If only there is a file data access requestfrom a user, do Cloud Servers re-encrypt the requested filesand update the requesting user's secret key. This statisticallysaves a lot of computation overhead since Cloud Servers areable to "aggregate" multiple update/re-encryption operationsinto one if there is no access request occurring across multiplesuccessive user revocation events.

File Access This is also the second stage of user revocation. Inthis operation, Cloud Servers respond user request on data fileaccess, and update user secret keys and re-encrypt requesteddata files if

necessary. As is depicted in Fig. 5, Cloud Serversfirst verify if the requesting user is a valid system user in*UL*. If true, they update this user's secret key componentsto the latest version and re-encrypt the *DEK*s of requesteddata files using the latest version of *PK*. Notably, CloudServers will not perform update/re-encryption if secret keycomponents/data files are already of the latest version. Finally,Cloud Servers send updated secret key components as well as cipher texts of the requested data files to the user. On receivingthe response from Cloud Servers, the user first verifies ifthe claimed version of each attribute is really newer thanthe current version he knows. For this purpose, he needs toverify the data owner's signatures on the attribute information(Including the version information) and the correspondingpublic key components, i.e., tuples of the form (j, T_j') in Fig. 5. If correct, the user further verifies if each secret keycomponent returned by Cloud Servers is correctly computed. He verifies this by computing a bilinear pairing between sk_j' and T_j' and comparing the result with that between the old sk_j and T_j that he possesses. If verification succeeds, he replaces each sk_j of his secret key with sk_j' and update T_j with T_j'. Finally,he decrypts data files by first calling *ADecrypt(P, SK,E)* to decrypt *DEK's* and then decrypting data files using *DEK's*.

File Deletion This operation can only be performed at therequest of the data owner. To delete a file, the data ownersends the file's unique *ID* along with his signature on this*ID* to Cloud Servers. If verification of the owner's signaturereturns true, Cloud Servers delete the data file.

2) Required algorithm

Algorithms required for the basic operation are *ASetup*, *AEncrypt*, *AKeyGen*, *ADecrypt*, *AUpdateAtt*, *AUpdateSK*,*AUpdateAtt*4*File*, and *AMinimalSet*. As the first four algorithms are justthe same as *Setup*, *Encryption*, *Key Generation*, and*Decryption* of the standard KP-ABE respectively, we focuson our implementation of the last four algorithms. Fig.6depicts two of the four algorithms[8].

AUpdateAtt This algorithm updates an attribute to a newversion by redefining its system master key and public key component. It also outputs a proxy re-encryption key between the old version and the new version of the attribute.

AUpdateAtt4 File This algorithm translates the ciphertext component of an attribute *i* of a file from an old version into the latest version[8]. It first checks the attribute history list of this attribute and locates the position of the old version. Thenit multiplies all the PRE keys between the old version and thelatest version and obtains a single PRE key. Finally it applythis single PRE key to the ciphertext component E_i and returns $E_i^{(n)}$which coincides with the latest definition of attribute *i.*

AUpdateAtt (*i, MK*)

randomly pick $t_i' \leftarrow^R \mathbb{Z}_p$;
compute $T_i' \leftarrow g^{t'i}$, and $rk_{i\leftrightarrow i'} \leftarrow t_i'/t_i$;
output t_i', T_i', and $rk_{i\leftrightarrow i'}$.

AUpdateAtt4File (i, E_i, AHL_i)
if*i* has the latest version, exit;
search $AHL_i$and locate the old version of *i*;
// assume the latest definition of *i* in *MK* is $t_{i\,(n)}$.
$rk_{i\leftrightarrow i}^{(n)} \leftarrow rk_{i\leftrightarrow i'} \cdot rk_{i'\leftrightarrow i''} \cdot \ldots \cdot rk_{i(n-1)\leftrightarrow i(n)} = t_{i(n)}/t_i$;
compute$E_i^{(n)} \leftarrow (E_i)^{rk_{i\leftrightarrow i}^{(n)}} = g^{ti^{(n)s}}$;
output$E_i^{(n)}$.

Fig. 6: Pseudo-code of algorithm AUpdateSK andAUpdateAtt4File

AUpdateSK This algorithm translates the secret key componentof attribute *i* in the user secret key *SK* from an oldversion into the latest version. Its implementation is similarto *AUpdateAtt4File* except that, in the last step it applies$(rk_{i\leftrightarrow i}^{(n)})-1$ to SK_i instead of $rk_{i\leftrightarrow i}^{(n)}$. This is because t_i is the denominator of the exponent part of SK_i while in E_i it is a numerator[8].

AMinimalSet This algorithm determines a minimal set ofattributes without which an access tree will never be satisfied.For this purpose, it constructs the conjunctive normal form(CNF) of the access tree, and returns attributes in the shortestclause of the CNF formula as the minimal attribute set[8].

VI. RELATED WORK

Existing work close to ours can be found in the areas of "shared cryptographic file systems" and "access control of outsourced data". In [6], Kallahalla et al proposed Plutus as a cryptographic file system to secure file storage on untrusted servers. Plutus groups a set of files with similar sharing attributes as a file-group and associates each file-group with a symmetric lockbox-key. Each file is encrypted using a unique file-blcok key which is further encrypted with the lockbox-key of the filegroup to which the file belongs. If the owner wants to share a file-group, he just delivers the corresponding lockbox-key to users. As the complexity of key management is proportional to the total number of file-groups, Plutus is not suitable for the case of fine-grained access control in which the number of possible "file-groups" could be huge.

In [13], Goh et al proposed SiRiUS which is layered over existing file systems such as NFS but provides end-to-end security. For the purpose of access control, SiRiUS attaches each file with a meta data file that contains the file's access control list (ACL), each entry of which is the encryption ofthe file's file encryption key (FEK) using the public key of an authorized user. The extension version of SiRiUS uses NNL broadcast encryption algorithm [15] to encrypt the FEK of each file instead of encrypting it with each individual user's public key. As the complexity of the user revocation solution in NNL is proportional to the number of revoked users, SiRiUS has the same complexity in terms of each meta data file's size and the encryption overhead, and thus is not scalable.

Ateniese et al [14] proposed a secure distributed storage scheme based on proxy re-encryption. Specifically, the data owner encrypts blocks of content with symmetric content keys. The content keys are all encrypted with a master public key, which can only be decrypted by the master private key kept by the data owner. The data owner uses his master private key and user's public key to generate proxy re-encryption keys, with which the semi-trusted server can then convert the ciphertext into that for a specific granted user and fulfill the task of access control enforcement.

In [7], Vimercati et al proposed a solution for securing data storage on untrusted servers based on key derivation methods [16]. In this proposed scheme, each file is encrypted with a symmetric key and each user is assigned a secret key. To grant the access privilege for a user, the owner creates corresponding public tokens from which, together with his secret key, the user is able to derive decryption keys of desired files.

VII. CONCLUSION

In this paper we propose a scheme to achieve fine grainedness, data confidentiality, and scalability of college data in cloud.Colleges can have their own separate cloud or they can get storage space from service providers such as Amazon's EC2 and S3, Google App Engine, and Microsoft Azure etc.. This reduces time and work load on a college record administration block and cost for storage.

ACKNOWLEDGMENT

This work was supported by the teaching staff of Department of computer science and engineering, Government engineering college Kushalnagar, Karnataka.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. USB-EECS-2009-28, Feb 2009.

[2] Amazon Web Services (AWS), online at http://aws. amazon.com.

[3] Google App Engine, Online at http://code.google.com/appengine/.

[4] Microsoft Azure, http://www.microsoft.com/azure/.

[5] Cloud Computing Security Consideration. http://www.dsd.gov.au/infosec/cloudsecurity.htm

[6] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Scalable secure file sharing on untrusted storage," in *Proc. of FAST'03*, 2003.

[7] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc. of VLDB'07*, 2007.

[8] Shucheng Yu, Cong Wang, KuiRen, and Wenjing Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing" at IEEE INFOCOM 2010.
[9] Qian Wang, Cong Wang, KuiRen, Wenjing Lou, Jin Li. "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing" 2011.
[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc.Of* CCS'06, 2006.
[11] S. Yu, K. Ren, W. Lou, and J. Li, "Defending against key abuse attacks in kp-abe enabled broadcast systems," in *Proc. of SECURECOMM'09*, 2009.
[12] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. of EUROCRYPT '98*, 1998.
[13] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing remote untrusted storage," in *Proc. of NDSS'03*, 2003.
[14] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage,"in *Proc. of NDSS'05*, 2005.
[15] D. Naor, M. Naor, and J. B. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proc. of CRYPTO'01*, 2001.
[16] M. Atallah, K. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in *Proc. of CCS'05*, 2005.

Authors

Yogeesh A.C obtained his Bachelor of Engineering in Computer Science from R.L Jalappa Institute of Technology and Master of Technology in Computer science and engineering from M.V Jayaraman College of engineering, Bangalore. He is currently working as an Assistant Professor, Department of Computer Science and Engineering, Government Engineering College, Kushalnagar, Kodagu, Karnataka, India. His areas of interest are Cloud Computing, Data Mining and Digital Image Processing.

Dilip B.R, studying in final year Bachelor of Engineering, Department of Computer science and engineering, Government Engineering College Kushalnagar, Kodagu, Karnataka, India. His areas of interest are Cloud Computing, Artificial Intelligence and Computer Network.

Chandra, studying in final year Bachelor of Engineering, Department of Computer science and engineering, Government Engineering College Kushalnagar, Kodagu, Karnataka, India.. His areas of interest are Cloud Computing, Mobile Computing and Computer Networks.

Divya S Abhyankar, studying in final year Bachelor of Engineering, Department of Computer science and engineering, Government Engineering College Kushalnagar, Kodagu, Karnataka, India. Her areas of interest are Cloud Computing, Artificial Intelligence.