An Effective Secure Group Sharing Structure in Public Cloud Computing

Prof.Pallavi Dhade[1] Rohini Pawar[2] Priyanka Sarje[3] Priyanka Satkar[4]
[1]Professor [2,3,4]Student
[1,2,3,4]Department of Computer Engineering
[1,2,3,4]Savitribai Phule Pune University

Abstract— With the prominence of gathering information sharing out in public cloud computing, the protection and security of gathering sharing information have ended up two noteworthy issues. The cloud supplier can't be dealt with as a trusted outsider due to its semi-trust nature, and in this way the conventional security models can't be direct summed up into cloud based gathering sharing structures. In this paper, we propose a novel secure gathering sharing structure for open cloud, which can viably exploit the cloud servers' assist however with having no delicate information being presented to assailants and the cloud supplier. The system consolidates intermediary signature, improved TGDH and intermediary re-encryption together into a convention. By applying the intermediary signature system, the gathering pioneer can successfully give the benefit of gathering administration to one or more picked gathering individuals. The upgraded TGDH plan empowers the gathering to arrange and redesign the gathering key sets with the assistance of cloud servers, which does not oblige the greater part of the gathering individuals been online constantly. By embracing intermediary re-encryption, most computationally concentrated operations can be appointed to cloud servers without uncovering any private data. Broad security and execution investigation demonstrates that our proposed plan is profoundly efficient and satisfies the security necessities for open cloud based secure gathering sharing.

Key words: Secure Group Sharing, Forward Secrecy, Backward Secrecy, Public Cloud Computing, Group Key Agreement

I. INTRODUCTION

Cloud computing is Internet ("cloud") based development and use of computer technology ("computing"). It is a style of computing in which dynamically scalable and often virtualization resources are provided as a service over the internet. One of the most useful or important fundamental services offered by cloud providers are data storage. Let us consider a practical data application. A company allows to its staffs in the same group to store and share files in the cloud. However, it also provides a significant risk to the confidentiality of those stored or shared files. Specifically, the cloud servers managed by cloud providers are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential, such as business plans. To preserve data privacy, a basic solution is to encrypt data files, and then upload the encrypted data into the cloud.

First, identity privacy is one of the most significant obstacles for the wide deployment of cloud computing. Without the guarantee of identity privacy, users may be unwilling to join in cloud computing systems because their real identities could be easily disclosed to cloud providers and attackers. On the other hand, unconditional identity privacy may incur the abuse of privacy. For example, a misbehaved staff can deceive others in the company by sharing false files without being traceable. Therefore, traceability, which enables the group manager (e.g., a company manager) to reveal the real identity of a user, is also highly desirable. Second, it is highly recommended that any member in a group should be able to fully enjoy the data storing and sharing services provided by the cloud, which is defined as the multiple-owner manner. Compared with the single-owner manner, where only the group manager can store and modify data in the cloud, the multiple-owner manner is more flexible in practical applications. More concretely, each user in the group is able to not only read data, but also modify his/her part of data in the entire data file shared by the company. Last but not least, groups are normally dynamic in practice, e.g., new staff participation and current employee revocation in a company. The changes of membership make secure data sharing extremely difficult. On one hand, the anonymous system challenges new granted users to learn the content of data files stored before their participation, because it is impossible for new granted users to contact with anonymous data owners, and obtain the corresponding decryption keys. On the other hand, an efficient membership revocation mechanism without updating the secret keys of the remaining users is also desired to minimize the complexity of key management. Several security schemes for data sharing an untrusted server have been proposed. In these approaches, data owners store the encrypted data files in untrusted storage and distribute the corresponding decryption.

In these approaches, data owners store the encrypted data files in untrusted storage and distribute the corresponding decryption keys only to authorized users. Thus, unauthorized users as well as storage servers cannot learn the content of the data files because they have no knowledge of the decryption keys

II. EXISTING SYSTEM

The efficiency of scheme relies on that there is high attribute variability between different files and high attribute variability between different users. The efficiency of the schemes depends on the assumption that Cloud Servers must be absolutely trusted. Otherwise, Cloud Servers can launch the collusion attack with some curious leaving group members has tried to realize an *ABE* and proxy re-encryption based data sharing scheme in mobile devices, which also has the problem mentioned. Diffe-rent from a general group member, a group administrator usually mandates more than one leaf node, and he/she knows all the secret keys of these leaf nodes. Therefore, when a group administrator leaves, another *GA* or *GL* should mandate all these leaf nodes, change the security keys, and update security information of the group including the group private key.

III. PROPOSED SYSTEM

Our framework in this paper aims to reduce the overhead for the involved parties, while alleviating the trustiness dependence of the semi-trusted cloud provider. The proposed scheme supports the updating of the group key pair whenever group members' joining or leaving happens, which transfers most of the computational complexity and communication overhead to Cloud Servers without leaking the privacy. Privilege of group management can be granted to any specific group member, which can be revoked at any time. Enhanced on the original *TGDH*, with the help of Cloud Servers, the proposed scheme enables the group to negotiate and update the group key pairs even though not all of the group members are online together. Additionally, there is another crucial design request in this scenario: any group member, including the group leader, can be temporary offline and become online again at any time We use Cloud Servers' aid based enhanced *TGDH* scheme to dynamical updating group key pair when there're group members leaving or joining the group. Even though not all the group members are online together, our scheme can still do well. In order to providing forward secrecy and backward secrecy, digital envelopes should be updated based on proxy re-encryption, which can delegate most of computing overhead to Cloud Servers without disclosing any security information. From the security and performance analysis, the proposed scheme can achieve the design goal, and keep a lower computational complexity and communication overhead in each group members' side.

A. *Advantage*

- Easy to Share files
- Member to Member File Sharing
- Provide files.

B. *System Architecture*

Fig. 1: An Example of Cloud based group Sharing Scenario

Fig. 1 for example, the normal group application scenario in cloud can be described as follows: The group leader opens up a sharing area in the cloud to form a group application.

Then, he/she grants the group members the right to implement data management. All the data in this group are available to all the group members, while they remain private towards the outsiders of the group including the cloud provider. The group leader can authorize some specific group members to help with the management of the group, and this privilege can also be revoked by the group leader. When a member leaves the group, he/she will lose the ability to download and read the shared data again.

Our framework in this paper aims to reduce the overhead for the involved parties, while alleviating the trustiness dependence of the semi-trusted cloud provider. Additionally, there is another crucial design request in this scenario: any group member, including the group leader, can be temporary offline and become online again at any time. Main contribution of this paper can be summarized as follows:

1) The proposed scheme supports the updating of the group key pair whenever group members' joining or leaving happens, which transfers most of the computational complexity and communication overhead to cloud servers without leaking the privacy.
2) Privilege of group management can be granted to any specific group member, which can be revoked at any time.
3) Enhanced on the original Tree-Based Group Diffie-Hellman (TGDH), with the help of cloud servers, the proposed scheme enables the group to negotiate and update the group key pairs even though not all of the group members are online together. Any offline group member can launch group key synchronization when he/she becomes online again in the next time.

C. *Module*

Case Study and Data Collection

- Group Leader
- Admin Authentication
- Group Member

IV. MODULE DESCRIPTION

A. *Case Study and Data Collection*

We consider a case study of a web-based collaboration application for evaluating performance. The application allows users to store, manage, and share documents and drawings related to large construction projects. The service composition required for this application includes: Firewall (x1), Intrusion Detection (x1), Load Balancer (x1), Web Server (x4), Application Server (x3), Database Server (x1), Database Reporting Server (x1), Email Server (x1), and Server Health Monitoring (x1). To meet these requirements, our objective is to find the best Cloud service composition

1) Group Leader

The group leader opens up a sharing area in the cloud to form a group application. Then, he/she grants the group members the right to implement data management. All the data in this group are available to all the group members, while they remain private towards the outsiders of the group including the cloud provider. The group leader can authorize some specific group members to help with the management of the group, and this privilege can also be revoked by the group leader. When a member leaves the group, he/she will lose the ability to download and read the shared data again.

a) File Upload

The group leader can upload the file for the group members. And the files are encrypted.

b) Re-Encrypt

The group leader should re-encrypt the members file.

c) Select Admin
The group leader can authorize some specific group members to help with the management of the group, and this privilege can also be evoked by the group leader.
d) Accept Request
The group leader also accepts the new member request.

2) Admin Authentication

The group leader can authorize some specific group members to help with the management of the group, and this privilege can also be evoked by the group leader. And the Admin can accept the new user request.

3) Group Member

Each group member can implement file download and upload operations in the authenticated group. Each *GM* can get some related public information from Cloud Servers and compute the specific set of security parameters, such as group key pair.
a) Share Data
The group members can share their data into another member in same group the data will translate by encrypted data.
b) Upload Data
The group members can upload the file to group leader. And the group leader can re-encrypt the data
c) Download File
The group members also download the group leader file.

4) Design Goals

In this section, we describe the main design goals of the proposed scheme including access control, data confidentiality, anonymity and traceability, and efficiency as follows:
a) Access Control:
The requirement of access control is to fold. First, group members are able to use the cloud resource for data operations. Second, unauthorized users cannot access the cloud resource at any time, and revoked users will be incapable of using the cloud again once they are revoked.
b) Data confidentiality:
Data confidentiality requires that unauthorized users including the cloud are incapable of learning the content of the stored data. An important and challenging issue for data confidentiality is to maintain its availability for dynamic groups. Specifically, new users should decrypt the data stored in the cloud before their participation, and revoked users are unable to decrypt the data moved into the cloud after the revocation.
c) Anonymity and traceability:
Anonymity guarantees that group members can access the cloud without revealing the real identity. Although anonymity represents an effective protection for user identity, it also poses a potential inside attack risk to the system. For example, an inside attacker may store and share a mendacious information to derive substantial benefit. Thus, to tackle the inside attack, the group manager should have the ability to reveal the real identities of data owners.
d) Efficiency:
The efficiency is defined as follows: Any group member can store and share data files with others in the group by the cloud. User revocation can be achieved without involving the remaining users. That is, the remaining users do not need to update their private keys or reencryption operations. New granted users can learn all the content data files stored before his participation without contacting with the data owner.

V. TECHNIQUE PRELIMINARIES

A. TGDH Based Group Key Agreement

The TGDH protocol in uses an adaptation of binary key trees in the context of fully distributed group key agreement based on Decisional Diffie-Hellman problem. Let p and q be two prime numbers which satisfy the condition q|p1 and the size of p and q are large enough so that solving the discrete logarithm problem in G is infeasible computational, where G is a subgroup with order q of a finite field Z*p . Let g be a generator of G. The binary key tree in TGDH pro-tocol is organized in the following manner: each node <l,v>is associated with a secret key K<l,v>and the corresponding blinded keyBK<l,v>= g^K<l,v>mod p. Each secret key K<l,v>of the internal node <l,v>is the Deffie-Hellman exchanged key between its two child nodes and can be computed recursively as follows:
K<l,v>=BK<l+1,2v+1>^K<l+1;2vi mod p
=BK<l+1,2vi>^K<l+1,2v+1>mod p
=g^K<l+1,2v^K<l+1,2v+1>mod p

The key pair at the root node (K(0,0) and BK(0,0)) is the established group key pair (group public key PuKGand group private key PrKG) shared by all group members:

PuKG=K(0,0)and PrKG=BK(0,0). Each group member is associated with a leaf node, whose security key is randomly and securely chosen.

Based on the TGDH protocol, Each group member Mi at the leaf node <l,v>knows all publicly shared blinded keys of sibling nodes of all nodes in the path from <l,v>to <0,0>and can compute all secret keys of nodes in the path. For example in Fig. 2, M2 knows his/her secret key K(3,1)and the blinded keys broadcasted by other group members:

Bk(3,0), Bk(2,1), Bk(1,1). Therefore, M2 can compute the key pairs of nodes (2,0),(1,0)and (0,0). There are five basic operations in TGDH: Join, Leave,Merge, Partition and Key-refresh. From we know that:

1) A joining operation requires two rounds (broadcast) with two messages. The number of modular exponentiations is O(2h-2)and O(h-1)(h =[log(n)]), where O(2h-2)modular exponentiations are needed by the sponsor to compute h 1 security keys Ks and blinded keys BKs, and O(h-1)modular exponentiations are needed by each other member to compute related updated key in his/her path from its associated node to the root node.
2) A leaving operation requires one round with one message. The number of modular exponentiation needed are also O(2h-2)and O(h-1).

There have been a lot of works to enhance the robustness of TGDH, including how to keep the stability when frequently joining and leaving, overhead optimization when more than one group members joining or leaving at the same time, and so on. However, all of these schemes do not con-sider how to do key negotiation when not all the group members online together at the sa-me time. The assumption that all group members should be online together cannot be guaranteed in the cloud environment,

which makes that the traditional TGDH is not suit-able. This paper will put forward an improved scheme to deal with this problem.

VI. OUR PROPOSED SCHEME

This section first gives an overview of our proposed scheme, then describes the scheme in detail which mainly consists of five phases: Group Initialization, Group Administration Privilege Management, Group Member Leaving and Joining (including Group Member Leaving, Group Member Joining and Group Administrator Leaving), Key Synchronizing, and Data Sharing Management.

A. Overview

Obtaining storage and computing resource from the cloud provider, the group leader GL implements the phase of Group Initialization to initialize a binary tree and some related security information of the group. Then GL can unicast the private key of each leaf node to the associated group member under the protection of encryption and signature. With the help of cloud servers' storage, each member can compute the group private key PrKG.

Relying on the proxy signature, the phase of Group Administration Privilege Management can help GL grant the group administration privilege to some specific group members. Furthermore, we divide the phase of Group Member Leaving and Joining into three possible sub-phases: Group Member Joining, Group Member Leaving and Group Administrator Leaving. Through the sub-phase of Group Member Joining, a group administrator and the new joining group member interact with each other to update security information of the group, including the group key pair PrKGand PuKG. Forward Secrecy should be guaranteed when a group member joins, which ensures that the newly joined user can also access and decrypt the previously published data. Therefore, all the old digital envelopes used to protect session

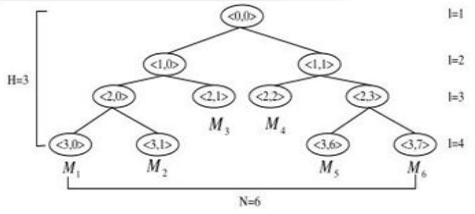

Fig. 2.A TGDH key Tree with six nodes.

Keys, which are generated to encrypted previously published data don't need to be updated. When a group member leaves, his/her associated node is mandated by a group administrator. In the sub-phase of Group Member Leaving, the group administrator GA launches enhanced TGDH based group key updating and then generates a proxy reencryption key from the version of group public key used in the existing digital envelopes to the new updated version. Different from a general group member, a group administrator usually mandates more than one leaf node, and he/ she knows all the secret keys of these leaf nodes. Therefore, when a group administrator leaves, another GA or GL should mandate all these leaf nodes, change the security keys, and update security information of the group including the group private key. The proxy re-encryption implementation is like that used in the sub-phase of Group Member Leaving. With the algorithm of proxy re-encryption, cloud servers can update all existing digital envelopes to be encrypted under the new updated group public key. Key Synchronizing is a key part of enhanced TGDH in our scheme. With the help of cloud servers, it makes temporarily offline group members can compute the current agreed group private key and other security information which needs to be synchronized.

The phase of Data Sharing Management describes the method how to securely upload and download file in the group. Furthermore, we detailed describe all these phases in the following sections.

B. Group Initialization

INITIALIZE()
1 Generate a shortest binary tree T with n leaf nodes.
▷ each node in T only have 0 or 2 children.
2 PrK_G = INIT_BINARY_T(ROOT)
▷ *Root* is the root node of T
3 $PuK_G = g^{PrK_G} \ mod \ p$
4 Associate each leaf node to a group member M_i, $i = 1, 2, ..., n$

INIT_BINARY_T(s)
1 **if** ($s.left$ && $s.right$)
2 **do** $ln = s.left$
3 $rn = s.right$
4 $ln.BK = g^{\text{INIT\_BINARY\_T}(ln)} \ mod \ p$
5 $K = ln.BK^{\text{INIT\_BINARY\_T}(rn)} \ mod \ p$
6 $ln.version = rn.version = 0$
7 **else**
8 **do** $K = random()$
9 $s.K = K$
10 $s.BK = g^{s.K} \ mod \ p$
11 **return** K

C. Group Administration Privilege Management

GAs can help the group leader GL manage the group, including accepting new group member's joining request, assisting group members to join the group and handling members' leaving event. GL can authorize and revoke the administration privilege to/from some specific group members by his/her will. When GL authorizes a group member GM j to be an GA, GL first sets the combination of some semantic information such as M j 's identity ID M j , the starting time, period of validity and the qualification of signing, etc. as the warrant information (m w M j) for the delegation to M j . M j obtains a pair of proxy signature keys PPrK M j and PPuK M j. After that, M j can be verified by any other personnonly if he/she knows GL's public key. If a signature signed by M j over specific data has passed the verification, M j can be considered as a legitimate proxy signer delegated by GL, and the verified data can be accepted by the verifier. There are also possibly leaving events for GA. We will describe the group member leaving and the group administrator leaving processes. From these two sections, we can find that the group administrator leaving process is more complex than the group member leaving process. Based on this reason, selecting group members to become group administrators should have the great probability to be online for a long time.

D. Group Member Joining

– Randomly select a security key.

- Get the blinded keys of all sibling nodes of every node in the path from his/her associated node to the root node from cloud servers.
- Compute new security keys and blinded keys of each node in the path from his/her associated node to the root node.
- Set the versions of his/her associated node and its parent node to "0". Add 1 to the version of each of the other internal nodes in this path.
- Send all the blinded keys from his/her associated node to the root node in this path to the GAj in an authentication tunnel.

E. Group Member Leaving

Implementation process is further described as follows:

```
MEM-LEAVING-PROC(leav_mem_n)
1  if leav_mem_n.sibing is also mandated by an GA
2        do Merge leav_mem_n, leav_mem_n.sibling,
             and leav_mem_n.parent to one node
3           Set this merged node
             as the current mandated node cur_mand_n
4     else Set leav_mem_n as cur_mand_n.
5  NODE_UPDATING_PROCESS(cur_mand_n)

NODE_UPDATING_PROC(n)
1  n.K = random( )
2  n.BK = g^{n.K} mod p
3  n.version = 0
4  r = n.sibling
5  s = n
6  for t is each node in the path
   from n.parent to the root node
7  if (t ≠ root)
8        do t.K = s.K^{r.BK} mod p
9           t.BK = g^{t.K} mod p
10          t.version++
11          s = t
12          r = s.sibling
13    else Exit;
```

F. Group Administrator Leaving

```
ADMIN-LEAVING-PROCESS(leav_admin)
1  Paths from each of leav_admin's mandated nodes
     and associated node to the root node
     form a subtree T'
2  UPDAT_SUBTREE(Root in T')

UPDAT_SUBTREE(n)
1  if n is not a leaf node in T'
2        do ln = n.left in T
3           rn = n.right in T
4           if ln is in T'
5                 do if rn is in T'
6                       do rn.BK =
                          g^{UPDAT_SUBTREE(rn)} mod p
7                 else      do get rn.BK
                             in T from Cloud Servers
8                    ln.K = UPDAT_SUBTREE(ln)
9                    K = rn.BK^{ln.K} mod p
10    else if ln is not in T' and rn in T'
11          do rn.K = UPDAT_SUBTREE(rn) mod p
12             rn.BK = g^{rn.K} mod p
13             get ln.BK in T from Cloud Servers
14             K = ln.BK^{rn.K} mod p
15    else if n is a leaf node in T'
16       do K = random()
17 n.K = K
18 n.BK = g^K mod p
19 return K
```

VII. KEY SYNCHRONIZING

When an offline member(taking M i for example) becomes online again, he/she should implement the key synchronizing process to get the current agreed group private key PrK G and the current group private key used in DEs(PrK DEG). M i gives the index of his/her ass-ociated leaf node and node version of every internal node in the path to cloud servers. Cloud servers first get the right position according to M i's given index. Because of the node joining process with the possible node splitting, the position with the given index may not be a leaf node. If this scenario happens, cloud severs search in the direction of left down along the left subtree, until reach a leaf node. This reached leaf node can be set as M i's current associated node.

Cloud servers reply the blinded keys of all siblings of every node in the path from M i's current associated leaf node to the root node. If the position with the given index is a leaf node, for each inherent node of the path from U i's associated node to the root node, cloud servers compare its version with its given versions from M i, until to a node when in equality circumstance happens(assume the index of this node is <I,j>). We can see that all the keys of the subbinary tree with root <i,j> haven't changed(except <i, j>).Cloud servers reply the blinded keys of all sibling nodes of every node in the path from the position with the index <i, j> to the root node.the group mem-ber M i can compute security keys and blinded keys of every node in the path. The se-curity key of the root node is the group priv-ate key PrK NEW, and the blinded key of the root node G. M i can verify whether the is the group public key PuK NEW G computed group public key is equal to the received group public key from cloud servers. Then, M i decrypts E PuK NEW PrK G DE Þ to get PrK G DE . When uploading files, he/G she uses PuK DE G to generate the related DEs. Meanwhile, after downloading files, he/she uses PrK DE G to decrypt the related DE to get the encryption keys.

VIII. DATA SHARING MANAGEMENT

Before uploading a file to cloud servers, the data owner gives the semantic description of the file: DESCRIPTION, which is convenient for searching in the group. Then, the data owner symmetrically encrypts the file with a randomly chosen session key SK. Together with uploading the encrypted sharing file, the data owner also uploads a digital envelope E PuK DE S (asymmetrically encrypt the session G key SK with the group public key PuK DE G), which is currently used in DE generation. Each file is stored in the cloud as the format:

$$ID||DESCRIPTION||\{File\}_{SK}||E_{PuK_G^{DE}}(SK)$$

Where ID is a unique identifier for the file. Group members online or ones from offline to online should timely get the updated E PuK NEW PrK G DE from cloud G servers to get the group private key used to decrypt DEs: PrK G DE. When a group member M i requests to download a file, he/she sends a request REQ to cloud servers. Cloud servers respond with a random number NUM and a signature:

Where NUM is a challenge code to M i, and PuK CS is the public key for cloud servers in the system. M i first verifies the signature. If passed, he/she uses the current agreed group private key PrK NEW to sign NUM: d= Sign

(NUM|| G NEW T M i; PrK G), where T M i is a timestamp value. M i sends d to cloud servers. Cloud servers verify the timeliness of T M i: whether the timestamp of the message is in a permitted time window. Then, cloud servers use the current agreed group public key PuK G NEW to verify the signature. If passed, they send the encrypted file and the specific digital envelope({File} SK ||E PuK DE (SK)) to M i . M i first G uses PrK G DE to get SK, and then decrypts {File} SK to get the request file.

$$NUM, Sign(H(NUM||REQ), PuK_{CS}),$$

IX. Performance Analysis

We discuss computational complexity and communication overhead in our scheme. In the process of group initialization, the group leader GL should give the definition of some necessary security parameters. GL also initializes TGDH-based binary tree and securely unicasts separate security keys to every group members, which contains O(N log2N)times exponential modular computation and O(N)times unicast communication.

These operations are one-time activities which are only implemented in the initialization stage. For granting the privilege of group administration to a specific group member, the group leader GL needs to create the warrant information mw and signs it, then securely transmits the signed mw to the specific group member. These operations contain two times asymmetric encryption and one time symmetric encryption. When a group member leaving or joining the group, the group leader GL or the specific group administrator who acts as a sponsor chooses a new security key for leaving group member's or new joining group member's associated leaf node, computes other related security keys and blinded keys, and then transmits all the computed blinded keys to cloud servers.

The transmitted message should contain a proxy signature provide authentication. All these operations contain O(log2N)exponential modular computation, one time proxy signing operation and one time communication. When there's a group member leaving the group, digital envelopes need to be updated based on proxy reencryption by cloud servers, which contains O(L)proxy reencryption computation(L is the number of sharing files in the group). When there's a group member joining the group, there is no need to update digital envelopes, and only a encryptedshould be computed, which contains only O(1)time encryption operation on the side of the group leader or the group administrator who acts as a sponsor.

When a group administrator (taking GAifor example) leaves from the group, another GA (GAj) should mandate GAi's mandated leaf nodes. GAjchooses new security keys for each of these leaf nodes, and computes security keys and blinded keys of every node in the paths from each of these leaf nodes to the root node. Then, he/she transmits new mandating information, all blinded keys and the proxy re-encryption key to cloud servers. The transmitted message should contain a proxy signature to provide authentication. All these operations contain (O(log2N),O(Nlog2N)) exponential modular computation, one time proxy signing operation and one time communication. In our scheme, we choose long-time online group members to become GA in order to prevent frequently launch GA leaving process. For key synchronization, the group member online or becoming offline to be online again timely publicly gets related blinded keys from cloud servers, and then computes security keys and blinded keys of each node in the path from his/her associated node to the root node. All these operations contains O(log2N)exponential modular computation and one time communication. Then the group member decrypts key, which contains one time exponential modular computation. For uploading a file, the file owner needs to choose a session key, encrypts the file, and generates a digital envelope. All these operations contain one symmetric encryption, one time asymmetric encryption, and one time communication.

The complexity of symmetric encryption and communication is linear with the length of the file. Before downloading a file, cloud servers should first verify whether the group member knows the current group private key to provide authentication. In our scheme we use a Challenge-Response game, containing two time communication and two times asymmetric encryption on downloading group member's side. After the verification, the downloading group member can get the file and the related digital envelopes from cloud servers, then decrypt the digital envelopes to get the session key and decrypt the encrypted file, which contains one time asymmetric encryption, one time symmetric encryption and one time communication to download the file. Here, the complexity of symmetric encryption and communication is linear with the length of the file.

X. Conclusion

In this paper we proposed a dynamic secure group sharing framework in public cloud computing environment. In our pro-posed scheme, the management privilege can be granted to some specific group me-mbers based on proxy signature scheme, all the sha-ring files are secured stored in cloud serv-ers and all the session key are protected in the digital envelopes. We use cloud servers' aid based enhanced TGDH scheme to dynamical updating group key pair when there're group members leaving or joining the group. Even though not all the group members are online together, our scheme can still do well. In ord-er to providing forward secrecy and backward secrecy, digital envelopes should be updated based on proxy re-encryption, which can de-legate most of computing overhead to cloud servers without disclosing any security info-rmation. From the security and performance analysis, the proposed scheme can achieve the design goal, and keep a lower comput-ational complexity and communication overhead in each group members' side.

References

[1] Security Issues and their Solution in Cloud Computing Prince JainMalwa Polytechnic College Faridkot, Punjab-151203, India prince12.jain@gmail.com
[2] A Study On Data Security Issues In Public Cloud Alycia Sebastian, Dr. L. Arockiam
[3] Enhanced Security System for Dynamic Group in Cloud V.Sathana #1 ,M.E-CSE Final Year,Info Institute of Engineering, Coimbatore Anna University, Chennai, India.
[4] A Survey of the Existing Security Issues in Cloud Computing ParulChachra

[5] Department of Computer Science, College of Vocational Studies University of Delhi, New Delhi.
[6] Three Level Security System for Dynamic Group in Cloud V.Sathana 1, J.Shanthini 2,1 M.E, Assisatant Professor 2, Dept of Computer Science and Engineering, Info Institute of Engineering, Coimbatore-India Anna University anusathana@gmail.com 1, shanthinisampath@gmail.com 2.
[7] M. Bellare, S. Keelveedhi, and T. Ristenpart.Dupless: Serveraided encryption for deduplicated storage. In USENIX Security Symposium, 2013.
[8] M. Bellare, S. Keelveedhi, and T. Ristenpart.Message-locked encryption and secure deduplication. In EUROCRYPT, pages 296– 312, 2013.
[9] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.

