International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 04 | Apr-2016 www.irjet.net p-ISSN: 2395-0072

Security Provisioning System for Group Communication in Public Cloud

Sayli kinage, Rupali Jahagirdar, Kajal Inamdar, Prachi Patil
Students, Dept. of Information Technology Engineering, MET'S BKC IOE Nasik, Maharashtra, India

---

Abstract - With the popularity of group data sharing in public cloud computing, the privacy and security of group sharing data have become two major issues. The cloud provider is semi-trust in nature, and thus the traditional security models cannot be directly used into cloud based group sharing frameworks. In this paper, we propose a secure group sharing framework for public cloud, which can effectively take advantage of the Cloud Servers' help but have no sensitive data being exposed to attackers and the cloud provider. The framework combines proxy signature, enhanced TGDH and proxy re-encryption together into a protocol. By applying the proxy signature technique, the group leader can effectively grant the privilege of group management to one or more chosen group members. The enhanced TGDH scheme enables the group to negotiate and update the group key pairs with the help of Cloud Servers, which does not require all of the group members been online all the time. By using proxy re-encryption, most computationally intensive operations can be given to Cloud Servers without disclosing any private information. Extensive security and performance analysis shows that our proposed scheme is highly efficient and satisfies the security requirements for public cloud based secure group sharing.

Key Words: **Forword Secrecy; Backword Secrecy;**

1. INTRODUCTION

The demand of outsourcing data has greatly increased. To satisfy the need for data storage and high performance computation, many cloud computing service providers have appeared, such as Amazon Simple Storage Service, Amazon S3 etc. There are two advantages to store data in Cloud Servers: 1) The data owners avoid the buying extra storage servers and hiring server management engineers; 2) It is easier for the data owner to share their data with intended recipients when the data is store in the cloud.

Despite of the advantages the privacy and security of users' data have become two major issues. the cloud is usually maintained and managed by a semi-trusted third party (Cloud provider).therefore it is desirable that data owner's credential information is in encrypted form and user can share there data to intended recipients.

The conventional approach to address the above mentioned problem is to use cryptographic encryption mechanisms, and store the encrypted data in the cloud. Authorized users can download the encrypted files and decrypt them with the given keys. There have been several other works on the privacy preserving data sharing issue in cloud based on various cryptographic tools, such as attribute based encryption (*ABE*), proxy re-encryption, etc. Among these existing schemes. have provided a fine-grained and scalable solution.

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 reduces the overhead of involved parties and it gives better performance while any group member including group leader temporarily offline and become online again.

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 *TGDH*, with the help of Cloud Servers.

2. RELATED WORK

2.1 Security

In Yu et al.'s scheme, an encrypted file can be decrypted by a user only if he/she has all of the file's attributes. By using proxy re-encryption, the computing complexity of digital envelope generation for a session key of a sharing file

decreases to only $O(1)$ at the data owner's side. For each one-time session key, the data owner needs to compute only one digital envelope by using his/her own public key. The efficiency of Yu et al.'s scheme relies on that there is high attribute variability between different files and high attribute variability between different users. But in group applications, different group members usually have same or similar interests, and they usually have attributes in common between them. In the scenario of interest based group sharing, if using Yu et al.'s scheme, the communication and computing overhead of user revocation will be dependent on the size of the group. So, in order to protecting files from the prying eyes of curious Cloud Servers and leaving group members, the data owner needs to regenerate is key pairs and regenerate $N-1$ proxy-reencryption keys when revoking a group member. In traditional studies, the security of group communication applications can be ensured by group key agreement, which can provide both *backward secrecy* and *forward secrecy*, which are not totally the same as that defined in cloud based group sharing. These schemes can be divided into two categories: centralized and distributed, all of which require all group members to be online together during the protocol implementation. Unfortunately, it's difficult to have such ―online together‖ guarantee in group applications in the cloud. How to make sure that such group applications in the cloud are secure and reliable remains a challenging problem. Although the scheme only requires asynchronous communication channels, it still requires the group members to participate in the process of protocol implementing and receive some others' sent messages when members' joining and/or leaving. Meanwhile, if a group member acting as a sponsor keeps in storing the private key of the shadow node, when he/she leaves the group, it is hard to keep backward secrecy in this scheme. Our work gives the extension to it to make more operability when any member online or offline at any time. In our scheme, based on Cloud Servers' help, Group members can implement key synchronization when they become online in the next time. We have also discussed the mode of security operations in cloud-based group applications.

3. MODULES

3.1 Network Model

Network model in this paper is shown in Fig. 1, where the group membership can change over time: each group member except the group leader can leave or apply to join the group at his/her will. Moreover, each group member in the group can be temporary offline and become online again at any time. Regardless of whether everyone is online or offline, the group can negotiate a group key pair (the group public key and the group private key) with the help of Cloud Servers. This group Key pair is used to protect the data shared in the group. Group members' leaving and joining can launch key updating process. Temporary offline group members should be also considered in protocol design. When these group members become online again, they should implement key synchronizing to compute to get the current key pair.

There are three kinds of users in cloud based group sharing applications:

1) ***Group Leader*** (*GL* in short): There is only one group Leader, who is the group creator and the top level group administrator. He/she buys or obtains storage and computing resource from the cloud provider. *GL* can authorize specific group members to manage the group, and this privilege of management can also be revoked by *GL*. *GL* provides initial group security parameters for all group members in the group.

2) ***Group Administrator*** (*GA* in short): There are 0, 1, or more authorized group administrators in a group. They can maintain group membership, and acts as sponsors to implement group key updating. Their privilege of management can be revoked by the group leader at any time. They also have all the functions of basic group members, such as uploading and downloading.

3) ***Group Member*** (*GM* in short): Each group member can implement file download and upload operations in the Authenticated group.

3.2 Security Model

The cloud provider is ***semi trusted***. We try to find out as much secret information as possible based on each group member's inputs. In general, we assume Cloud Servers are interested in data contents and group member's security information rather than other secret information. Our scheme should satisfy the security requirements of *backward secrecy and forward secrecy*. The former one ensures that the revoked user cannot decrypt new cipher texts and the newly joined user can also access and decrypt the previously published data. We assume that an adversary can be a passive attacker who could be a man-inthe-middle to monitor the communications among the group members and Cloud Servers. A former group member can communicate with Cloud Servers and try to access data contents shared in his/her former group. An active adversary is able to impersonate an legitimate group member to gain some right. In general, we say that our scheme is secure if no adversary can succeed with any possible attacks mentioned above.

4. TECHNIQUES

4.1 Proxy Signature

Proxy signature is a signature scheme, in which an original signer can delegate his/her signing capability to a proxy signer, and then the proxy signer generates a signature on behalf of the original signer. From a proxy signature, a

verifier can be convinced of the original signer's agreement on the signed message. There are various methods but we use Proxy signature *with warrant*, which is proved to be more secure and practical, so we also use partial delegation with warrant in our protocol design.

Let „A' be an **original signer** who has an authentic key pair(*PrKA* and *PuKA*) & „B' be a **proxy signer** who has an authentic key pair(*PrKB and PuKB*). The *mw* be A's warrant information for the delegation.δA = *Sign*(*PrKA;mw*) be A's signature on the warrant *mw* using his/her private key PrKA. A transmits δA to the proxy signer B.Then partial delegation with warrant based proxy signature scheme is described as follows:

o (***Proxy signature key generation***) *PKG* is a proxy

signature key generating algorithm that takes original signer's signature δA and proxy signer's private key *PrKB* as inputs, and outputs a proxy signature key pair(*PPrKB*, *PPuKB*). It is executed by the proxy signer:

(*PPrKB; PPuKB*) ← *PKG*(δA*; PrKB*) (1)

o (***Proxy signing***) *PS* is a proxy signing algorithm

that takes proxy signature private key *PPrKB* and message *m* as inputs, and outputs proxy signature δP . It is executed by the proxy signer *B*:

δP ← *PS*(*m; PPrKB*) (2)

o (***Proxy signature verifying***) *PSV* is a proxy signature verifying algorithm that takes (δP, *m*, *mw*, *PuKA*, *PuKB*) as inputs, and outputs either accept or reject. It is executed by any verifier:

PSV(δP*; m;mw; PuKA; PuKB*) ?= *accept or reject* (3)

4.2 *TGDH* based Group Key Agreement

The *TGDH* protocol uses an adaptation of binary key trees in the context of fully distributed group key agreement based on Decisional Diffie-Hellman problem. The binary key tree in *TGDH* protocol is organized in the following manner: each node ⟨*l; v*⟩ is associated with a secret key *K*⟨*l;v*⟩ and the corresponding blinded key *BK*⟨*l;v*⟩ = *gK*⟨*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;2*v*+1⟩*K*⟨*l*+1;2*v*⟩ *mod p*

=*BK*⟨*l*+1; 2v⟩*K*⟨*l*+1; 2v*+1*⟩ *mod p*

=*gK*⟨*l*+1; 2v⟩*K*⟨*l*+1;2*v+1*⟩ *mod p* (4)

The key pair at the root node (*K*⟨ 0;0⟩ and *BK*⟨ 0;0⟩) is the established group key pair (group public key *PuKG* and 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 & securely chosen.

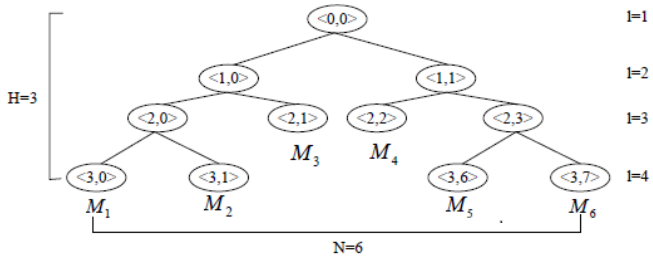

Fig. 2. A *TGDH* key Tree with 6 nodes

For example in Fig. 2, *M*2 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, *M*2 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*.

1) A joining operation requires two rounds (broadcast) with two messages. The number of modular exponentiations is $O(2h-2)$ and $O(h-1)(h = \lceil \log(n) \rceil)$, where $O(2h-2)$ modular exponentiations are needed by the sponsor to compute $h-1$ security keys *K*s and blinded keys *BK*s, 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 work 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 consider how to do key negotiation when not all the group members online together at the same time. The assumption that all group members should be online together cannot be guaranteed in the cloud environment, which make that the traditional *TGDH* is not suitable. This paper will put forward an improved scheme to deal with this problem.

4.3 Proxy Re-encryption

Proxy re-encryption is an cryptographic primitive in which one person (Take the user *A* for example) allows a semitrusted proxy to re-encrypt message that will be sent to another designated person (Take the user *B* for example). *A* should generate a proxy re-encryption key *rkPuKA→PuKB* by combining his/her secret key with *B*'s public key. This re-encryption key is used by the proxy as input of the re-encryption function, which is executed to convert a ciphertext encrypted under *A*'s public key (*PuKA*) into another ciphertext that can be decrypted by *B*'s private key (*PrKB*). Except for converting, the proxy cannot see the underlying data contents.

Proxy re-encryption is extensively used to provide ciphertext updating in cloud environment. By this way, most

computational intensive operations of ciphertext updating can be transferred to Cloud Servers, without reveal any content of ciphertext to them.

5. OUR PROPOSED SCHEME

This section first gives an overview of our proposed scheme, 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*.

5.1 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 *PrKG* and *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 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 re-encryption 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 mandate

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.

6. SECURITY ANALYSIS

◦ ***Certificateless Authentication***. Based on proxy signature, *GL* can grant the privilege of group administration to some group members as *GA*s. *GAi* only needs to provide *mwGAi* and *PuKGAi* , everyone who knows *GL*'s public key can verify whether *GAi* has got *GL*'s authorization.

◦ ***Backward Secrecy When a Group Member Leaves.*** This is provably secure based on the hard Decisional Bilinear Diffie-Hellman problem. When a group member leaves, there position in the binary tree is mandated by a GA or the group leader *GL*. As illustrated in Fig.3, a GM only knowing the security key of one leaf node could compute security keys of every node in the path from the leaf node to the root node. Because of this reason, the security key of the mandated node should be changed after the group member's leaving. In our scheme, the security key and blinded key of each node in the path from the mandated node to the root node can be updated, so the group key pair is changed to a new one. Because leaving group member cannot know the new security key of his/her previously associated node and any other leaf nodes' security key, he/she cannot compute the updated group key pair.

◦ ***Backward Secrecy When a Group Administrator Leaves.*** This is also provably secure based on Diffie-Hellman problem. When a *GA* leaves, all his/her mandated and associated nodes should be mandated by another *GA*. In order to make the leaving *GA* computing the final group key pair becomes impossible, all security keys of these nodes should be changed by the new mandator. After the group administrator leaving process, all other group members still in the group can compute the final updated group key pair by requesting some necessary updated blinded keys from Cloud Servers. However, the leaving *GA* cannot know any leaf node's security keys, so he/she cannot compute the updated group key pair.

◦ ***Cloud Provider Cannot Compute the Group Private Key***. Although the cloud provider knows all the blinded keys of every node in the binary tree, he/she cannot know any leaf nodes' security keys. he/she cannot compute the security keys of any internal nodes and the root node, so he/she cannot

get the final group private key.

◦ ***Data Confidentiality***. Just as in traditional ways, each file shared in the group (*FILE*) is symmetric encrypted with a session key(*KEY*): *{FILE}KEY* and *KEY* is asymmetric

encrypted with the receiver's public key *PuK*:*EPuK*(*KEY*). Assume the symmetric encryption algorithm and asymmetric encryption algorithms are secure, suppose *KEY* is encrypted with the group public key *PuKG*. Now the security relies on *PrKG*'s security, so the scheme should guarantee only authenticated group members know the current group private key. Therefore, the security of *PrKG* can be guaranteed. Also, when there's a group member leaving the group, *PrKG* can be timely updated. Meanwhile Cloud Servers use re-encryption to change the digital envelopes from being under previous group public key to be under the new group public key. Only current group members who know the new group private key can decrypt the download file.

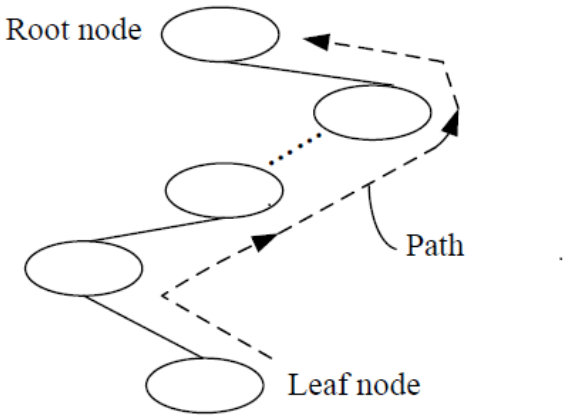

Fig. 3. Path illustration from a leaf node to the root node

7. CONCLUSION

A dynamic secure group sharing framework in public cloud computing environment is proposed by this system. The management privilege can be granted to some specific group members based on proxy signature scheme, all the sharing files are secured stored in Cloud Servers and all the session key are protected in the digital envelopes. System use Cloud Servers aid based enhanced TGDH scheme to dynamical updating group key pair when there are group members leaving or joining the group. Even though not all the group members are online together, given 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
There are no sources in the current document.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.

ACKNOWLEDGMENT

Every work is source which requires support from many people and areas. It gives me proud privilege to complete the project work on —Security Provisioning System For Group Communication In Public Cloud " under valuable guidance and encouragement of our guide Prof. Priti Lahane I am extremely grateful to Prof. Namita Kale (Head of Department) and Prof. Priti Lahane(Project Guide) and Dr. Kalpana Metre(Project Co-ordinator) for providing all facilities and every help for smooth progress of project work. At last I would like to thank all the staff members and my friends who directly or indirectly supported me without which the project work would not have been completed successfully.

REFERENCES

[1] M. Blaze, G. Bleumer, and M. Strauss,*" protocols and atomic proxy cryptography, in Advances in Cryptology EUROCRYPT ",* International Conference on the Theory and Applications of Cryptographic Techniques Proceedings 1998.
[2] M. G. G. Ateniese, K. Fu and S. Hohenberger, —Improved proxy re-encryption schemes with applications to secure distributed storage,‖ *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.
[3] S. Kim, S. Park, and D. Won, —Proxy signatures, revisited‖ *Information and Communications Security*, pp. 223–232, 1997.
[4] Rui Zhang,PeiShuai Chen, "A Dynamic Cryptographic Access Control Scheme in Cloud Storage Services", *College of Computer Science In-formation Engineering,*
Zhejiang Gongs hang University,, 2006.
[5] I.Luan, A.Muhammad, P.Milan, *An encryption scheme for a secure policy updating, In: International Conference on Security and Cryp-tography,* SECRYPT 2010, Athens,Greece, pp. 399-408,2010.
[6] J.Bethencourt, A.Sahai, B. Wwters, *Ciphertext-policy attributed-based encryption, In: IEEE Symposium on Security and Privacy*. Berkeley, California, USA, pp. 321-334, 2007.
[8] B. Lee, H. Kim, and K. Kim, —Secure mobile agent using strong non-designated proxy signature,‖ in *ACISP2001: Proc. 6th Australasian Conference on Information Security and Privacy*, vol. 2119, 2001, pp. 474–486.