International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 5; Issue: 7; July -2016

Group Sharing Framework with Dynamic Security and Privacy in Public Cloud Computing

[1]Prathibha T C, PG Scholar, Dept. of CSE, BNMIT, Bengaluru
[2]Chandrakala H T, Assistant Professor, Dept. of CSE, BNMIT, Bengaluru

Abstract: With the increased need for data storage and high performance computation, storing of data and creating private group in the public cloud space became advantageous. It saved the data owners from the trouble of buying extra storage servers and also hiring server management engineers. Security and privacy are the two major issues. Cloud service provider is a semi-trusted third party, traditional security models like firewalls, routers etc. cannot be used to meet security and privacy concerns. Hence cryptographic mechanisms are used. This required updating of secret key pairs whenever group member leaving or joining the group in order to provide backward and forward secrecy. Therefore a novel secure framework combines TGDH (Tree based Group Diffie-Hellman) scheme, Key Synchronization, proxy signature, proxy re-encryption and data sharing among classes together in the protocol. The TGDH facilitates secret key pairs updating with reduced computational complexity. This required all the group members to be online all the time which was not practical. Any offline member can implement the key synchronization process to get the updated group key pairs when he becomes online. To reduce the load of group management on the group leader privilege of group management can be granted to the group administrator using proxy signature. Proxy re-encryption re-encrypts the data with the help of proxy. To provide higher level of privacy data sharing among classes is used.
Keywords-Group sharing framework, public cloud computing, backward and forward secrecy

1. Introduction

Safeguarding data and secure sharing with the intended recipient is the greatest challenge. Data is very precious, once if it is lost or corrupted can never be regained. The data owners need to buy some extra storage servers maintain backups and hire management engineers which is not cost efficient. If same file has to be shared with number of recipients sending one after another increased the delay time. Hence creating a group in the cloud space and sharing data among the group members resulted in reduced delay time and also saved the cloud space.

A group is created in the cloud space with people of similar interest for files and the file is shared in the group. For example shown in figure 1 group leader creates a group in the cloud space and permits access of the cloud space to all the group members. Every group member can upload and download files. Group leader is the manager of the group and the load of group management is more on the group leader. So group leader provides group management privilege to one or more trusted group member called the group administrator. The group administrator manages the group in spite of the group leader. Private group is created in the public cloud space. File access is provided only to the authorized members in the group and the file remains invisible to the outsiders. If any group member leaves he loses the access to the files in the group and if any group member joins he must be able to access the files uploaded before he joined into the group. Hence group key updating is required when group member joins or leaves the group.

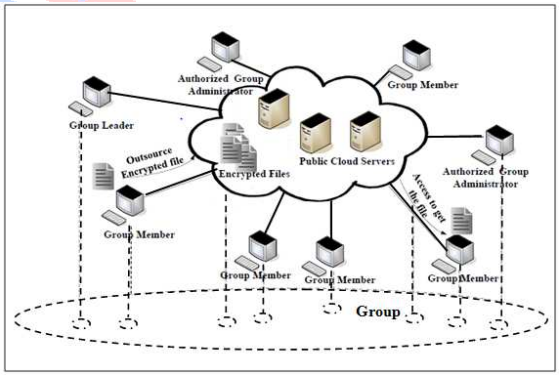

Figure 1 private group created in the public cloud space

Despite of the above advantages of storing data in the cloud there arrived some considerable challenges, among which privacy and security are the two major issues. The data owner cannot fully rely on the cloud service provider, as cloud service provider is a semi-trusted third party. Hence measures must be taken to preserve data from both the cloud service provider as well as the attacker. The attacker may misuse the data and the cloud service provider itself may leak the sensitive data stored in the cloud space. To meet this purpose traditional security models like firewalls, routers etc. cannot be used as they can easily trespassed. Hence cryptographic mechanisms are used to store the encrypted data in the cloud space. This required secure sharing of secret key which is required to decrypt the encrypted file at the receiver side. The even more challenging problem was to share this secret key securely with the intended recipient.

2. Literature survey

Initially digital envelope [1] was used to address the task of secure group secret key exchange. In the digital envelope approach the data was encrypted with the session key using symmetric encryption and the session key was encrypted with the destination's public key using asymmetric encryption and was shared in the cloud space. For example if user A is the sender and user B is the receiver, user A wants to randomly choose a session key and encrypt the data using symmetric encryption algorithm such as DES or AES and then encrypts the session key with destination's public key using asymmetric encryption algorithm such as RSA. The session key encrypted with the destination's public key is called the digital envelope. The user A shares both the file and the digital envelope in the cloud space and user B downloads both, extracts the session key and decrypts the file. In this approach if the file is shared with N specific users then N digital envelopes are required and also if there are M files to be shared then M more such N digital envelopes are required. Therefore the overall overhead is O(MN) in the digital envelope approach.

Hybrid attribute-and re-encryption based key management[2] has done several 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, Yu et al. have provided a fine-grained and scalable solution. The efficiency of this scheme relies on that there is high attribute variability between different files and high attribute variability between different users. The efficiency of the early 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. This has tried to realize an *ABE* and proxy re-encryption based data sharing scheme in mobile devices, which also has the problem mentioned in earlier schemes.

DAC-MACS [3] scheme satisfied the security requirements of backward secrecy and forward secrecy. The former one ensures that the revoked user cannot decrypt new cipher texts. The later one ensures that the newly joined user can also access and decrypt the previously published data. This two security requirements are usually used in some cloud based data sharing scenarios

A potential adversary may be a former group member or any one out of the group. We assume that an adversary can be a passive attacker who could be a man-in the- middle to monitor the communications among the group members and Cloud Servers. A former group member can collude with Cloud Servers and try to access data contents shared in his/her former group. An active adversary is able to impersonate a legitimate group member to gain some right.

Tree-based group key agreement [4] made a preliminary attempt, which provides a fully distributed TGDH (Tree-Based Group Diffie-Hellman) based scheme. 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. This scheme did not work on giving the extension to it to make more operability when any member online or offline at any time.

The decision diffie-hellman problem [5] scheme is also provably secure based on the hard Decisional Bilinear 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 mandatory. 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.

In Yu et al.'s scheme [6], 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. Based on the proxy re-encryption algorithm, Cloud Servers can compute digital envelopes for all intended recipient. 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.

3. Proposed System

The proposed scheme is mainly designed to reduce the overall overhead caused during group secret key pairs updating when group member leaves or joins the group. It mainly consists of six phases: Group initialization, Group administration privilege management, Group member leaving and joining, Group administrator leaving, key synchronization and data sharing management.

3.1 Overview

The group leader obtains the storage and computing resource from the cloud provider, implements the Group Initialization phase and creates a private group in the cloud space. In the group initialization phase the binary tree of the associated group members is created and all the group members are provided access to the cloud space.

By Group administration privilege management phase the group leader can effectively grant the privilege of group management to one or more trusted group member through

proxy signature to reduce the load of group management on him.

When Group member leaving and joining happens all the group secret key pairs has to be updated to provide backward and forward secrecy. When a group member leaves he is no more the member of the group and must lose access to the files in the group, for this purpose group key pairs has to be updated which is called the backward secrecy. When a group member newly joins the group he must be able to access the encrypted files uploaded before he joined the group, for this purpose again the group key pairs has to be updated which is called as forward secrecy. TGDH based group key updating is used to meet this purpose. The group administrator mandates the key updating process using TGDH scheme. Proxy re-encryption is also used to provide higher level of security taking the help of trusted third party without exposing data to him.

If Group administrator leaving leaves even then group key pairs need to be updated because the group admin knows the group key pairs of all the mandated group members. Therefore if the group admin leaves another group admin or the group leader must mandate the group key pairs updating process and update keys of all members mandated by the leaving group admin.

But this TGDH scheme required all the group members to be online all the time which is not practical. Hence key synchronization is used to overcome this drawback. Any offline member can implement the key synchronization process the next time he comes online to get the updated group key pairs.

Data sharing management phase involves methods to securely upload and download the file in the group. All the phases are described in detail in the subsections.

3.2 **Group Initialization**

After obtaining storage and computing resource from the cloud provider, *GL* generates a shortest binary tree with *n* leaf nodes, where *n* is the number of group members (including *GL* itself) in the initial group. Each node of this binary tree can only have either zero (as a leaf node) or two child nodes, which means a node with one child is not allowed. Each one of these *n* leaf node is mandated by GA found Set found node as associated joining member associated with one different group member.

Pseudo code:

1. GL initializes group.
2. GL assigns grp_admin.
3. Registers some grp_mems.
4. Gen Usr_name and psswrd to mems.
5. Send login credentials to every member.
6. If of usr_name and psswrd matches
7. Enable login
8. Else exit
9. Gen Grp_keys (TGDH)
10. Compute: $PuKt = {}^{PrKt}modq$
11. Send to all grp_mem with signature (HMAC_SHA1 algorithm).

3.3 **Group Administration Privilege Management**

*GA*s 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 *GMj* to be an *GA*

3.4 **Group Member Joining**

When a group member joins, he/she sends a joining request to one group administrator (taking *GAj* for example). *GAj* handles this joining event as a sponsor. After verifying the new joining group member's legitimacy.

Pseudo code for Group member joining:

1. Request to join to GA
2. If joining member's legitimacy verified
3. Joining member's process
4. Else exit

Joining member's process (n)

1. If Jn_mem_n has to join the group, its position is mandated by GA
2. Key_updation_proc(n)
3. Broadcast keys to every member

Key_updation_proc(cur_mand_node_n)

1. For n is number of users
2. If(n=! 0)
3. PrKt= random()
4. Compute: $PuKt = {}^{PrKt}modq$
5. Else exit

3.5 **Group Member Leaving**

When a group member leaves, in order to provide backward secrecy, the group key pair should be updated, and all digital envelopes related to the sharing data in this group should be also updated and encrypted by the new group public key. In our scheme, one *GA* should mandate leaving group member's position and act as a sponsor to implement the group member leaving process.

Pseudo code for Group Administrator leaving:

1. When GA quits, another GA or GL should take care of leaving member's position
2. Key_updation_proc(n)
3. Broadcast keys to every member

Key_updation_proc(cur_mand_node_n)

1. For n is number of users
2. If(n=! 0)
3. PrKt= random()
4. Compute: $PuKt = {}^{PrKt}modq$
5. Else exit

3.6 **Group Administrator Leaving**

Usually each *GA* mandates more than one leaf node, and he/she knows the secret keys of these leaf nodes. When an *GA* leaves, another *GA* or *GL* should mandate these leaf nodes and change the security keys instead of him/her.

Pseudo code for Group Administrator leaving:

1. When GA quits, another GA or GL should take care of leaving member's position
2. Key_updation_proc(n)
3. Broadcast keys to every member

Key_updation_proc(cur_mand_node_n)

1. For n is number of users
2. If(n=! 0)
3. PrKt= random()
4. Compute: $PuKt = {}^{PrKt}modq$
5. Else exit

3.7 Key Synchronizing

When an offline member (taking *Mi* for example) becomes online again, he/she should implement the key synchronizing process to get the current agreed group private key *PrKG* and the current group private key used in *DE*s (*PrKDEG*).

3.8 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*.

4. Techniques

I. TGDH based Group Key Agreement:

General public elements:

q Prime number

α primitive root of q

User A key Generation

Select private key Xa Xa < q

Calculate public key Ya $Ya = \alpha^{Xa} modq$

User A key Generation

Select private key Xb Xb < q

Calculate public key Yb $Yb = \alpha^{Xb} modq$

Secret key generation by user A $K = (Yb)^{Xa} modq$

Secret key generation by user b $K = (Ya)^{Xb} modq$

II. Proxy Signature

Proxy signature is a signature scheme, in which an original signer can delegate user signing capability to a proxy signer, and then the proxy signer generates a signature on behalf of the original signer proxy signature scheme is described as follows:

1) (**Proxy signature key generation) PKG**

(PPrKB, PPuKB) ← PKG(δA, PrKB)

2) (**Proxy signing) PS**

δP ← PS(m, PPrKB)

3) (**Proxy signature verifying) PSV**

PSV (δP, m, mw, PuKA, PuKB)?= accept or reject

III. Proxy Re-encryption

Proxy re-encryption is an cryptographic primitive in which one person allows a semi-trusted proxy to re-encrypt user message that will be sent to another designated person.

$rk_{PuKa \rightarrow PuKb}$

IV. Key Synchronizing

When an offline member becomes online again, he/she should implement the key synchronizing process to get the current agreed group private key PrKG and the current group private key used in DEs (PrKG DE).

V. Sharing of data among classes

To provide higher level of privacy sharing of data between only administrator and team leader or team leader and group member etc is done.

5. Snapshots

Figure 2 Overview of Client portal

Figure 3 Overview of Group Leader portal window

Figure 4 Overview of Group Admin portal window

Figure 5 Overview of Group Member portal window

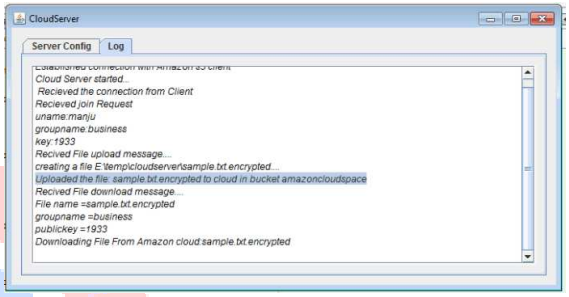

Figure 6 Server side GUI log

Figure 6 Files stored in Amazon s3 cloud

6. Conclusion

Hence in the proposed scheme secure group key updating is done. Group management privilege is provided to group administrator using proxy signature. Proxy re-encryption is used to provide higher level of security. Hence the overall overhead is also successfully reduced.

References

[1] W. Jia, H. Zhu, Z. Cao, L. Wei, and X. Lin, "SDSM: A secure data service mechanism in mobile cloud computing," in WKSHPS2011: Proc. 2011 IEEE Conference on Computer Communications Workshops. IEEE CS, 2011, pp. 1060–1065.

[2] P. Tysowski and M. Hasan, "Hybrid attribute-and re-encryption based key management for secure and scalable mobile applications in clouds," IEEE Transactions on Cloud Computing, vol. 1,no. 2, pp. 172–186, 2013.

[3] "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in Proceedings of IEEE INFOCOM 2013 satisfied the security requirements of backward secrecy and forward secrecy.

[4] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement,"ACM Transactions on Information and System Security (TISSEC), vol. 7, no. 1, pp. 60–96, 2004.

[5] D. Boneh, The decision diffie-hellman problem. Springer, 1998, pp. 48–63.

[6] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable and fine-grained data access control in cloud computing," in IEEE INFOCOM 2010: Proc, the 29th Conference on Computer Communication, IEEE, 2010.

[7] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in Advances in Cryptology- EUROCRYPT '98, Proceedings of International Conference on the Theory and Application of Cryptographic Techniques, Springer-Verlag, 1998, pp. 127–44.

[8] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," IEEE-ACM Transactions on Networking, vol. 8, no. 1, pp. 16–30, 2000.

[9] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 8, pp. 769–780, 2000.

[10] W. Yu, Y. Sun, and K. R. Liu, "Optimizing the rekeying cost for contributory group key agreement schemes," IEEE Transactions on Dependable and Secure Computing, vol. 4, no. 3, pp. 228–242, 2007.

[11] A. Perrig, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in CCS'00: Proc. 7th ACM conference on Computer and communications security. ACM, 2000, pp. 235–244.

[12] T. Jung, X. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in Proceedings of IEEE INFOCOM2013, IEEE, 2013, pp. 2625–2633.