Cloud storage with improved access control and assured deletion

Ranjana Badre
Department of Computer Engineering
MIT Academy of Engineering, Alandi, Pune, India

Abstract- Providing secure and efficient access to large scale outsourced data is an important issue of cloud computing. In this paper, a mechanism FADE, a secure overlay cloud storage system, which will guarantee assured file deletion and improved access control for outsourced data is proposed. Cryptographic approach is used for storing and managing data..

Keywords – Cloud storage, Access control, Assured deletion, Backup/recovery

I. INTRODUCTION

Providing robust data to users is an important and difficult task for outsourced data providers. On the other hand, in some cases it is required for data to become unrecoverable in reliable manner after some time. For example, health records are required to destroy to protect patient privacy, or corporations might have policies about how long email should be retained. Though cloud storage is an attractive, the security of outsourced data has become most important issue now a days. One major challenge is to provide guarantee of *assured deletion* .i.e., data files are permanently inaccessible upon requests of deletion. It is undesirable to keep data backups permanently, as there is possibility of exposing sensitive information in the future because of data breach or erroneous management of cloud operators. Thus, to avoid liabilities, enterprises and government agencies usually keep their backups for a finite number of years and request to delete (or destroy) the backups afterwards.

The aim of Assured deletion is to provide cloud clients an option of reliably destroying their data backups upon requests. On the other hand, cloud providers may replicate multiple copies of data over the cloud infrastructure for fault-tolerance reasons. Cloud clients do not know how many copies of their data are on the cloud, or where these copies are located since cloud providers do not publicize their replication policies. It is unclear whether cloud providers can reliably remove all replicated copies when cloud clients issue requests of deletion for their outsourced data.

Thus, it is necessary to design a highly secure cloud system that enables assured deletion for outsourced data backups on the cloud, while addressing the important feature of access control.

II. RELATED WORK

A. *Assured Deletion*

Assured deletion can be achieved by multiple ways. One of the approaches is *secure overwriting* [1].In this approach, original data is overwritten with new data to make original data unrecoverable. Overwriting large numbers of blocks is exceedingly time-consuming and is rarely adopted as a regular operation. It requires internal modification of a file system. Therefore the technique is not applicable for outsourced data because backends are maintained by third parties and there is no guarantee that replicated data will be overwritten.

Second approach is *disk scrubbing* [2].Here system delete data on disk by overwriting it many times. The data on the disk might be unrecoverable after multiple over writings but there is no guarantee that all backup copies of data will be simultaneously destroyed.

Self-destruct[3] is a feature of email systems. This system assures that copy of email at the client side will be assuredly deleted after reading. Again there is no use of cryptography. So backup copies of data will remain in the storage.

Another approach is, use of cryptography. Here the cryptographic keys those are used for decrypting data are removed to make encrypted data unrecoverable. In this approach data is stored in outsourced storage while keys are kept and maintained at key manager [4],[5],[6],[7],[8],[9].FADE [8] supports policy based assured deletion, in which data can be assuredly deleted according to revoked policies. FADE[10] is asecure overlay cloud storage system that achieves fine-grained, policy-based access control and file assured deletion.

Boneha and Lipton[11] proposed two implementation scenarios for Assured Deletion.1)The expiration time is known at file creation.2)On-Demand deletion of individual files. The on-demand scheme is very scalable. However, compared with the predetermined expiration time scheme there is a potential usability issue.

B. Access Control

For enabling secure and efficient access to outsourced data, investigators have tried to integrate key derivation mechanisms [12],[13],[14],[15] with encryption based data access control. Atallah et al. [16] proposed a method that uses only hash functions to derive a descendant's key in a hierarchy. This method can handle updates locally and avoid propagation. The proposed key derivation tree structure can be viewed as a special case of access hierarchies. In [17], the authors created groups of users based on their access rights to the data. The users are then organized into a hierarchy and further transformed to a tree structure to reduce the number of encryption keys. The advantage of this method is it helps to reduce the number of keys that are given to each user during the initiation procedure. In [18], data records are organized into groups dependent on the users that can access them. Here changes to user access rights will results into updates in data organization because the data in the same group are encrypted by the same key. An innovative idea in this approach is to allow servers to conduct a second level encryption
(over- encryption) to control access, repeated access revocation and grant may lead to a very complicated hierarchy structure for key management. The approach used in [19], stores multiple copies of the same data record encrypted by different keys. Here when access rights change, reencryption and data up- dates to the server must be conducted simultaneously which leads to extra overhead on the server.

III. IMPLEMENTATION DETAILS

A System Design

Fig.1 gives an overview of our project. The cloud hosts data files on behalf of a group of users who want to outsource data files to the cloud depending on their definitions of file access policies. FADE is an overlay system atop the underlying cloud. It guarantees security protection to the outsourced data files before they are hosted on the cloud.

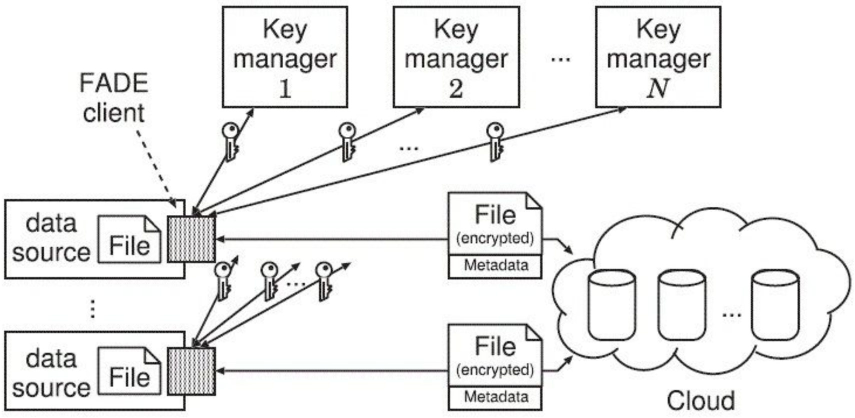

Fig 1 The architecture if the system

B Modules

This project has four modules.
1. Data Owner Module:
The data owner is the entity that originates file data to be stored on the cloud. It may be a file system of a PC, a user-level program, a mobile device, or even in the form of a plug- in of a client application. The data owner requests the key manager to decrypt a blinded version of the encrypted data key. If the associated policy is satisfied, then the key manager will decrypt and return the blinded version of the original data key. The data owner can then recover the data key. In this way, the actual content of the data key remains confidential to the key manager as well as to any attacker that sniffs the communication between the data owner and the key manager.

2. Key Manager Module:
The key manager maintains the policy-based control keys that are used to encrypt data keys. It responds to the data owner's requests by performing encryption, decryption, renewal, and revocation to the control keys. The key manager can be deployed as a minimally trusted third-party service. By minimally trusted, we mean that the key manager reliably removes the control keys of revoked policies. However, it is possible that the key manager can be compromised. In this case, an attacker can recover the files that are associated with existing active policies. On the other hand, files that are associated with revoked policies still remain inaccessible, as the control keys are removed. Hence, file assured deletion is achieved.
3. Storage Cloud (Third party provider) Module:
The storage cloud is maintained by a third-party cloud provider (e.g., Amazon S3) and keeps the data on behalf of the data owner. There is no requirement of any protocol and implementation changes on the storage cloud to support this system.
4. Policy Revocation for File Assured Deletion Module:
If a policy P_i is revoked, then the key manager completely removes the private control key d_i and the secret prime numbers p_i and q_i. Thus, we cannot recover S_i from S_i^{ei}, and hence cannot recover K and file F. Thus the file F, which is tied to policy P_i, is assuredly deleted. Hence the policy revocation operations do not involve interactions with the cloud.

C Access Control with Attribute based encryption
To recover a file from the cloud, a client needs to request the key manager to decrypt the data key (assuming that only a single key manager is deployed).The client needs to present authentication credentials to the key manager to show that it indeed satisfies the policies associated with the files. One implementation approach for this authentication process is based on the public-key infrastructure. However, this client-based authentication requires the key manager to have accesses to the association of every client and its satisfied policies. This limits the scalability and flexibility if we scale up the number of supported clients and their associations with policies.
To resolve the scalability issue, attribute-based encryption [20], [21], [22] turns out to be the most appropriate solution. In particular, The approach used here is based on Cipher text-Policy Attribute-Based Encryption (CP-ABE) [20].
In this approach, each client first obtains an ABE-based private access key from the key issuing authority of the ABE system that corresponds to a set of attributes the client satisfies. This can be done by having the client present authentication credentials to the key issuing authority, but it is emphasized that this authentication is only a one-time bootstrap process. Later, when a client requests the key manager to decrypt the data key of a file on the cloud, the key manager will encrypt the response messages using the ABE-based public access key that corresponds to the combination of policies associated with the file. If the client indeed satisfies the policy combination, then it can use its ABE-based private access key to recover the data key. Note that the key manager does not have to know exactly each individual client who requests decryption of a data key.
FADE uses two independent keys for each policy. The first one is the private control key that is maintained by the key manager for assured deletion. If the control key is removed from the key manager, then the client cannot recover the files associated with the corresponding policy. Another one is the ABE-based access key that is used for access control. The ABE-based private access key is distributed to the clients who satisfy the corresponding policy, as in the ABE approach, while the key manager holds the ABE-based public access key and uses it to encrypt the response messages returned to the clients. The use of the two sets of keys for the same policy enables FADE to achieve both access control and assured deletion. Now FADE operations can be modified to include the ABE feature as follows: Here it is assumed that we operate on a file that is associated with a single policy.
1. **File upload:** The file upload operation remains unchanged, since we only need the public parameters from the key manager for this operation, and hence we do not need to authenticate the client.
2. File download: The file download operation requires authentication of the client. When the client requests the key manager to decrypt $S_i^{ei} R^{ei}$, the key manager encrypts its answer S_iR with ABE based on the policy of the file. Therefore, if the client satisfies the policy, then it can decrypt the response message and get S_iR.
3. **Policy renewal:** Similar to above, the key manager encrypts S_iR with ABE when the client requests it to decrypt the old policy. For the re-encryption with the new policy, there is no need to enforce access control since we only need the public parameters.
4. **Policy revocation:** Here challenge-response mechanism is used in order for the key manager to authenticate the client. In the first round, the client tells the key manager that it wants to revoke policy P_i. The key manager then generates a random number r as a challenge, encrypts it with ABE that corresponds to policy P_i, and gives it to the

client. Next, if the client is genuine, then it can decrypt r and send its hash to the key manager as the response to that challenge. Finally, the key manager revokes the policy and acknowledges the client.

D Multiple Key Managers

The use of a single key manager will lead to the single point- of-failure problem. An untrustworthy key manager may either prematurely removes the keys before the client requests to revoke them, or fail to remove the keys when it is requested to. The former case may prevent the client from getting its dat back, while the latter case may subvert assured deletion. Therefore, it is important to improve the robustness of the key management service to minimize its chance of being compromised. Here, Shamirs (M,N) threshold secret sharing scheme is applied [23], where M <= N . Using Shamir's scheme, a secret is divided into N shares and distributed them over N independent key managers, such that the correct shares must be obtained from at least M out of N key managers in order to reconstruct the original secret.

In FADE, we need to address the challenge of how to manage the control keys with N >1 key managers. For each policy Pi, the jth key manager (where 1 <=j<= N) will independently generate and maintain an RSA public/private control key pair ($e_{i\,j}$, d_{ij}) corresponding to a modulus n_{ij} . This key pair is independent of the key pairs generated by other key managers, although all such key pairs correspond to the same policy P_i. Also, each key manager keeps its own key pair and will not release it to other key managers.

Let us consider a file F that is associated with policy P_i. The operations here are follows:

File upload

Instead of storing S_i^{ei} on the cloud as in the case of using a single key manager, the client now splits S_i into N shares, S_{i1}, S_{i2}, ...S_{iN} using Shamir's scheme. Next, the client requests each key manager j for the public control key (n_{ij} , e_{ij}).Then the client computes S_i^{ei} (mod n_{ij}) for each j, and sends {K}$_{Si}$, S_{i1}^{ei1} , S_{i2}^{ei2} ,……. S_{iN}^{eiN} and {F}$_k$ to the cloud. Finally, the client discards K, S_i and S_{i1} S_{i2}, ...S_{iN} .

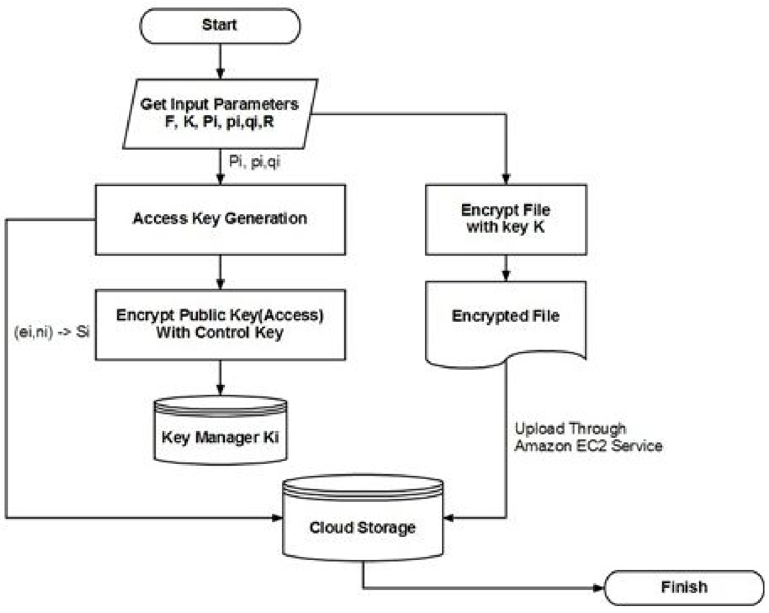

Fig. 2. Uploading Process

File download

After retrieving the encrypted key shares S_{i1}^{ei1} , S_{i2}^{ei2} ,……. S_{iN}^{eiN} from the cloud, the client needs to request each key manager to decrypt a share. For the jth share $S_{eij\ ij}$ (j = 1, 2, ...,N), the client blinds it with a randomly generated number R, and sends S_{ij}^{eij} R^{eij} to key manager j. Then, key manager j responds the client with $S_{ij}R$. It also encrypts the response with ABE. After unbinding, the client knows S_{ij}. After collecting M decrypted shares of S_{ij} , the client can combine them into S, and hence decrypts K and F.

Policy Renewal

The policy renewal operation is similar to original operation. The only difference is that the client needs to renew every share of S_i. In this operation there is no need to combine or split the shares.

Policy revocation
The client needs to ask every key manager to revoke the policy. As long as at least (N-M+1) key managers remove the private control keys corresponding to the policy, all files associated with this policy become assuredly deleted.

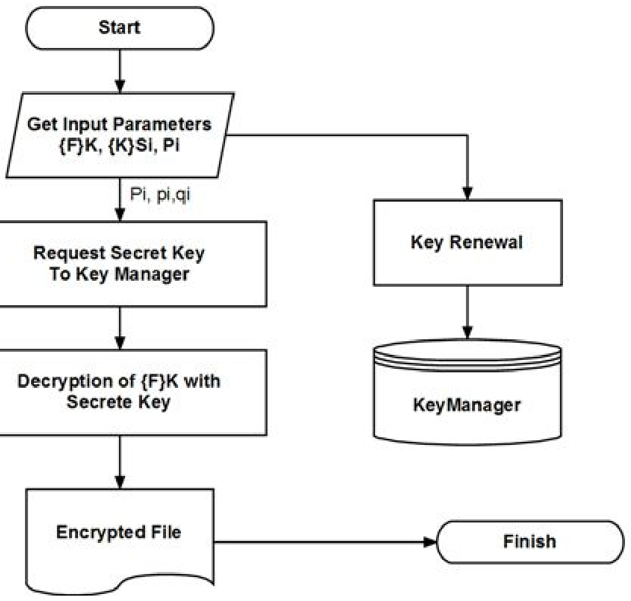
Start
Get Input Parameters {F}K, {K}Si, Pi
Pi, pi,qi
Request Secret Key To Key Manager
Decryption of {F}K with Secrete Key
Encrypted File
Finish
Key Renewal
KeyManager

Fig. 3. Downloading Process

IV.CONCLUSION

In this paper basic FADE architecture is discussed. Then extensions to FADE are given. Extended FADE is more suitable for enforcing security of outsourced data in the cloud. It guarantees Access control and Assured deletion to the data stored on the third party cloud.

REFERENCES

[1] P. Gutmann 8ecure deletion of data from magnetic and solid state memory. *In Proc. of USENIX Security Symposium*, 1996
[2] Junping Liu, Ke Zhou, Liping Pang, Zhilun Wang, Yuhui Deng and Den Feng , A novel cost effective scrubbing scheme. *In fifth International joint conferenceon INC, ISM and ISC 2009.*
[3] Lingfang Zeng, Shibin Chen, Qingsong Wei and Dan Feng, SeDas: A self-destructing data system based on active storage framework. *In APMRC, 2012, Digest.*
[4] D. Bonesh and R. Lipton. A revocable Backup System. *In Proc. Of USENIX Security Symposium, 1996.*
[5] R. Geambasu, J.P. John, S.D. Gribble. T. Kohno, and H. M. Levy, Keypad; An Auditing File System for Theft-Prone Devices. *In Proc. of ACM EuroSyz, 2011.*
[6] R. Geambasu, T. Kohan, A. Levy, and H. Levy. Vanish: Increasing data privacy with self-destructing data. *In Proc. of USENIX Security Symposium 2009.*
[7] R. Perlman, File System Design with Assured Delete.. *In ISOC NDSS, 2007*
[8] Y. Tang, P. Lee, J. Lui, and R. Perlman, FADE: Secure Overlay Cloud Storage with File Assured Deletion. *In Proc. of SecureComm, 2010*
[9] S. Yu, C. Wang, K. Ren and W. Lou, attribute Based data Sharing with Attribute Revocation. I ACM Symposium on Information, Computer and Communication Security (ASIACCS), Apr 2010.
[10] Yang Tang, Patrick P. C. John C. S. Lui and Radia Perlman . Secure Overlay Cloud Storage with Access Control and Assured Deletion. *In IEEE transactions on dependable and secure computing Vol. No. 6, December 2012.*
[11] Boneh D. and Lipton R., A Recoverable Backup System. *In Usentx Security Symposium, 1996.*
[12] T. Chen, Y. Chung, and C. Tian. A novel key management scheme for dynamic access control in a user hierarchy. *In IEEE Annual International Computer Software and Applications Conference, pages 396-401, 2004.*
[13] H. Chien and J Jan. New hierarchical assignment without public key *cryptography Computers, Security, 22(6): 523-526, 2003.*
[14] C. Lin. Hierarchical key assignment without public key cryptography, *Computers Security, 20(7): 612-619, 2001.*
[15] S. Zhong, A practical key management scheme for access control in a user hierarchy. *Computers Security, 21(8): 750-759, 2002.*
[16] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst Secur, 12(3): 1-43, 2009.*

[17] E. Damiani, S. D. C. di Vimercati, S Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Key management for muti-user encrypted databases. *In Proceedings of the ACM workshop on Storage security and survivability, pages 74-83, 2005.*

[18] 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 Proceedings of the international conference on Very large data bases, pages 123- 134, 2007.*

[19] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. A data outsourcing architecture combining cryptography and access control. *In Proceedings of the ACM workshop on Computer security architecture, pages 63-69, 2007, international conference on Very large data bases, pages 123-134, 2007.*

[20] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-Policy Attribute-Based Encryption, *Proc. IEEE Symp. Security and Privacy, May 2006.*

[21] V. Goyal, O. Pandey, A. Sahai, and B. Waters, Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data, *Proc. 13th ACM Conf. Computer and Comm.Security (CCS)*, 2006.

[22] A. Sahai and B. Waters, Fuzzy Identity-Based Encryption, *Proc. EUROCRYPT, 2005.*

[23] A. Shamir, How to Share a Secret, *Comm. ACM, vol 22, no. 11, pp. 612-613, Nov. 1979.*