Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

*IJCSMC, Vol. 2, Issue. 6, June 2013, pg.307 – 313*

RESEARCH ARTICLE

# A SECURE CLOUD WITH ADDITIONAL LAYER OF PROTECTION AND USER AUTHENTICATION

**Mr. Iranna Telasang[1], Mr. Manjunatha S[2]**
[1]IV SEM, M.Tech CNE, Department of CSE CMRIT, Bangalore, India
[2]Associate Prof, Department of CSE CMRIT, Bangalore, India

*[1] irannatelsang@gmail.com; [2] manju02@gmail.com*

***Abstract— Cloud storage is a model of networked online storage where data is stored in virtualized pools of storage which are generally hosted by third parties. Security is considered to be one of the most critical aspects in a cloud computing environment due to the sensitive and important information stored in the cloud for users. In order to protect data in cloud we use a secured system which is an extension of FADE (File Assured Deletion) system proposed earlier. In FADE system we associates policy to each file whenever it is stored into the cloud and assured deletion makes the file completely unrecoverable upon the revocation of the policy associated with the file. All keys related to the file encryption and decryption operations are self-maintained by a quorum of key managers that are independent of third-party clouds, with this we also encrypt the file with long term secret key in order to provide an additional layer of protection for the file and also we authenticate every user who request for the access to file. We implement this new FADE version and conduct evaluation atop windows azure cloud to demonstrate that FADE provides security protection for outsourced data.***

***In this paper, we would like to overcome weaker areas of FADE and design an even more security and also provide solution for reducing the overhead in FADE.***

***Key Terms: - Access control; Policy-based File Assured Deletion; cloud storage***

## I. INTRODUCTION

Cloud computing is now the hot spot of computer business and research., as it offers an abstraction of infinite storage space for clients to host data backups in a pay-as you use manner. It helps enterprises and government agencies significantly reduce their financial overhead. Since they can now archive their data backups remotely to third-party cloud storage providers rather than maintain data centres on their own. For example, Smug Mug [15], allow user to share their snaps in website, on Amazon S3 in 2006 and saved thousands of dollars on maintaining storage devices [3]. More case studies of using cloud storage for remote backup can be found in [4]. Apart from enterprises and government companies, each person can also store their personal data to the cloud using tools like Drop-box [6]. In particular, with the advent of smart phones, we expect that more people will use Dropbox like tools to move audio/video files from their smart phones to the cloud, given that smart phones typically have limited storage resources.

However, security concerns become relevant as we now outsource the storage of possibly sensitive data to third parties. In this paper, we are particularly interested in providing guarantees of limited access control, in which we ensure that only authorized users can access the stored data on the cloud. And we also provide

 

guarantees of assured deletion, that user data stored is permanently inaccessible to anybody (including the data owner) once we requests for deletion of data. Keeping data permanently may lead to unexpectedly disclosed in the future due to attacks on the cloud or careless management of cloud operators. Assured deletion is that we have to trust cloud storage providers to actually delete data is the challenge that should be achieved, but they may be reluctant in doing so [14]. Also, cloud storage providers typically keep multiple backup copies of data for backup. It can't be say that cloud service provider remove all backup copies upon requests of deletion.

In FADE, active data files that remain on the cloud are associated with a set of user-defined file access policies (e.g., time expiration, read/write permissions), such that data files are accessible only to users who satisfy the file access policies. In addition, FADE make universally applicable time-based file assured deletion [10] (i.e., data files are assuredly deleted upon time expiration) into a more fine-grained approach called policy-based file assured deletion, assuredly deletes the data files when the associated file access policies are revoked and become obsolete. The design of FADE easily tells that encrypted data remains on third-party (un-trusted) cloud storage providers and cryptographic keys are independently stored and maintained quorum of key managers. To assure the guarantees of access control and assured deletion, FADE uses cryptographic schemes including threshold secret sharing [9] and attribute-based encryption, [5] [7] [12] [13]and performs various cryptographic key operations that provide security protection for basic file upload/download operations. We design a proof of concept prototype of FADE to justify its feasibility, as a value-added security service, to enhance the properties of security of general data outsourcing applications.

We point out that the design of FADE is based on the thin-cloud interface [17], meaning that it only requires the cloud to support the basic data access operations. Therefore FADE is applicable for general types of storage backend, as long as such back ends provide the interface for uploading and downloading data. In cloud storage we need to specifically consider the performance metrics that are inherent to cloud storage which include data transmission performance (given that a cloud is deployed over the Internet) and monetary cost overhead (given that a cloud charges clients for data outsourcing). Thus, we evaluate FADE empirically according to the specific features of cloud storage, so as to prove that FADE is actually a feasible solution for secure cloud storage.

In this paper, we present a secure overlay cloud storage system which ensures file assured deletion and works seamlessly atop today's cloud storage services. FADE separates the management of encrypted data and encryption keys, such that encrypted data stored on third-party (un-trusted) cloud storage providers, while encryption keys are which are not dependent is maintained by a key manager service, which stores key in a quorum scheme [9]. FADE generalizes time-based file assured deletion (i.e., files are assuredly deleted upon time expiration) into a more fine-grained approach called policy based file assured deletion, in which files are associated with access policies (e.g., time expiration, read/write permissions of authorized users) and are assuredly deleted when the associated file access policies are revoked and become obsolete.

## II. BACKGROUND AND RELATED WORK

The previous Fade Version [1] follows the standard backup version-controlled design, which eliminates the storage of redundant data across different versions of backups. Fade Version applies cryptographic protection to data backups. Specifically, fine-grained assured deletion is enabled in which cloud clients can assuredly delete particular backup versions or files on the cloud and make them permanently cannot be accessed to anyone, while other versions that share the common data of the deleted versions or files will remain unaffected. This system has more time overhead as it has to check for copy of the data in cloud every time while uploading new data.

Coming to the assured deletion part, there are different ways of achieving assured deletion. One approach is by secure overwriting [2], in which old data is made unrecoverable by over writing on it. Secure overwriting has also been applied in versioning file systems [8]. However, this requires internal modifications of a file system and is not feasible for outsourced storage, since the storage back ends are maintained by third parties, and it has no guarantee that replicated data will be over written.

Time-based file assured deletion, which is first introduced in [10], means that files can be securely deleted and remain permanently inaccessible after a pre-defined duration. The idea is that a file by owner data key should be encrypted, and this data key is further encrypted with a control key by a separate key manager, here the key manager is a server that is responsible for cryptographic key management. In [10] the control key is time-based, meaning that it will be deleted by the key manager when it reaches expiration time, the expiration time is specified when the file is first declared. Without the control key and data key the data file remain encrypted and inaccessible. Therefore the main security property of file assured deletion is that even if a cloud provider does not remove expired file copies from its storage, then those files remain encrypted and unrecoverable.

An open issue in the work [10] is that it is uncertain that time-based file assured deletion is feasible in practice, as empirical evaluation will not be there. Later, the idea of time-based file assured deletion other forms are developed in Vanish [12]. Vanish divides a data key into multiple key shares which are then stored in

 

different nodes of a public Peer-to-Peer Distributed Hash Table (P2P DHT) system. Nodes remove the key shares which reside in their caches after a fixed time period. If a file needs to remain accessible even after the time period, the key shares in node caches should be updated by file owner. Vanish is built on the cache-aging mechanism in the P2P DHT, it is difficult to analyse the idea from time-based deletion to a fine-grained control of assured deletion with respect to different file access policies. In the following section we analyse this issue.

**Policy Based Deletion**

We now generalize time-based deletion to policy-based deletion. We represent each file with a single atomic file access policy (or policy for short), or more generally, a combinational boolean of atomic policies. Each (atomic) policy is associated with a control key, and the key manager maintains all control keys. Suppose now that a file is associated with a single policy. The file content is encrypted with a data key similar to time based deletion, and the data key is encrypted with the control key corresponding to the policy. When the policy is revoked, the corresponding control key will be removed from the key manager. The encrypted content of the file cannot be recovered with the control key of the policy, when the policy associated with a file is revoked and no longer holds the data key, we say the file is successfully deleted. To delete files that are associated with revoked policies is the main idea of policy-based deletion.

## III. PROPOSED SYSTEM

We now overview the design of FADE, guarantees of access control and assured deletion for outsourced data in cloud storage is provided by the system. The necessary components of FADE, and state the design and security goals that it seeks to achieve will be presented.

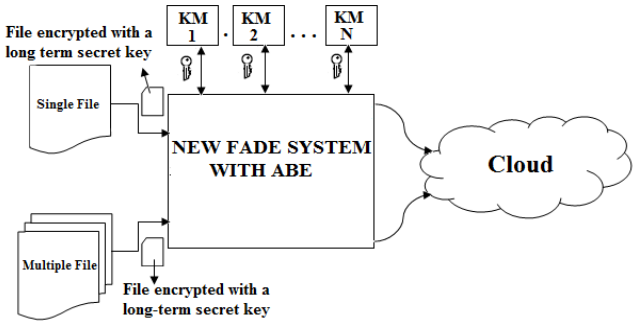


**Figure.1 Architecture of New FADE**

Figure 1 illustrates an overview of the FADE system. On behalf of a group of FADE users the cloud hosts the data files who want to outsource data files to the cloud based on their definitions of file access policies. FADE will be viewed as an overlay system atop the underlying cloud. security protection is provided to the outsourced data files before they are hosted on the cloud.

### 3.1 Entities

As shown in Figure 1, the FADE system is composed of two main entities:

• **FADE Client.** A FADE client *(*or client for short) is an interface that bridges the data owner (e.g., file system) and the cloud. It applies encryption (decryption) to the outsourced data files uploaded to (downloaded from) the cloud. To perform the necessary cryptographic key operations interacts with key manager.

• **Key manager.** FADE is built on a quorum of key managers [9], each of which is a stand-alone entity that maintains policy-based keys for access control and assured deletion.

The cloud, third-party provider, provides storage space for hosting data files on behalf of different FADE clients in a pay-as-you-go manner and is maintained by third party provider. Access policies are associated with individual data files. On the thin-cloud interface the FADE will be built [17], and assumes only the basic cloud operations for uploading and downloading data files. We give prominence that we do not require any protocol and implementation changes on the cloud to support FADE.

 

### 3.2 Cryptographic Keys

The three types of cryptographic keys will be defined by FADE to protect data files stored on the cloud:

• **Data key.** A data key is a random secret that is generated and maintained by a FADE client. FADE client is used for encrypting or decrypting data files via symmetric-key encryption (e.g., AES).

• **Control key.** A particular policy is associated with a control key. A public-private key pair is represented, and the private control key is maintained by the quorum of key managers. It is used to encrypt/decrypt the data keys of the files protected with the same policy. The basis of policy-based assured deletion will be formed by control key.

• **Access key.** Sane as control key, an access key is associated with a particular policy, and is represented by a public-private key pair. The FADE client that is authorized to access files of associated policy will maintain the private access key unlike the control key. Access key will be built on attribute-based encryption [5], and forms the basis of policy-based access control. Intuitively, to successfully decrypt an encrypted file stored on the cloud, we require the correct data key, control key, and access key. It is computationally infeasible to recover an outsourced file being protected by FADE without any of these keys.

• **Long term secret key.** The long term secret key is the key used to encrypt the file and the decryption of the same can be used when the key managers collude with each other and makes the file unrecoverable, hence in such a situation the owner of the file can decrypt the file by using this long term secret key, this key will be emailed to the owner once he upload his file.

## IV. FADE DESIGN

In this section, we present the design of FADE. We propose several cryptographic key operations in particular that enable FADE to achieve our security goals.

### 4.1 Basic Operations of FADE

We start with the basic design of FADE. We make two assumptions to simplify our discussion. First, a single key manager is used. Second, before a file is accessed, authentication credentials has to be presented by client (e.g., based on public key infrastructure certificates) to the key manager to show that it satisfies the proper policies associated with the files, cryptographic key operations will be performed by key manager.

### 4.2 File Upload/Download

We now introduce the basic operations of how a client uploads/downloads files to/from the cloud. Each file is associated with a single policy we initially start with this case. Our design is based on blinded RSA [16] (or blinded decryption [10]), in which the client requests the key manager to decrypt a blinded version of the encrypted data key. If the associated policy is satisfied, the blinded version of the original data key will decrypted and returned, after this client can then recover the data key. 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 client and the key manager this will be the main motivation of blinded decryption.

#### 4.2.1 File Upload

In the file upload operation the client first encrypt the file with ABE algorithm by attaching policies to it than this file is again encrypted AES algorithm, as a third layer of encryption we apply RSA to this finally encrypted file and generate public-private keys, we encrypt file with public key and then upload the file to the cloud. The private key S is squared (S*S) then sent to the key manager for splitting and storing into different key managers. The upload operation also involves encrypting of the file with long term secret key and sending that key to the owner of the file through E-mail.

#### 4.2.2 File Download.

In the file download operation the client fetches {K}S, $S^2$, and {F}K from the cloud. Then the client requests key managers for the minimum number of shares, once he get the shares ($S^2$), he has to perform square root operation to get original key S. Then client can decrypt {K}S and hence {F}K. The file can also be downloaded by the owner by using long-term secret key which has been mailed to him at the time of uploading the file.

#### 4.2.3 Policy Revocation for File Assured Deletion

If a policy $P_i$ is revoked, then the key manager completely removes the private control key S and the secret prime numbers $p_i$ and $q_i$. Thus, we cannot recover $S_i$, and hence cannot recover K and file F. From this we ensure that file F, which is tied to policy $P_i$, is assuredly deleted. The policy revocation operations do not involve interactions with the cloud is noticed.

#### 4.2.4 Multiple Policies

FADE supports a Boolean combination of multiple policies. We mainly focus on two kinds of logical connectives: (i) the conjunction (AND), means the data is accessible only when every policy is satisfied; and (ii) the disjunction (OR), means if any policy is satisfied, then the data is accessible. Following operations on a

 

Boolean combination of policies are similar to those in [11], while the focus of [11] is on digital rights management rather than file assured deletion.

**4.2.5 Policy Renewal**

We conclude this section with the discussion of policy renewal. Associating a file with a new policy which is nothing but policy renewal For example, if a user wants to extend the access of a file, then the user can update the old policy to the new policy that specifies a new access rule.

In FADE, without retrieving the encrypted file from the cloud the policy renewal operates the keys. The procedures can be summarized by following steps: (i) download all encrypted keys (including the data key for the file and the set of control keys for the associated policy) from the cloud, (ii) send them to the key manager for decryption, (iii) recover the data key, (iv) re-encrypt the data key with the control keys of the new policy, and finally (v) send the newly encrypted keys back to the cloud, (vi) e-mail new long-term secret key to the file owner.

**4.3 Access Control with ABE**

To recover a file from the cloud, to decrypt a client key a client needs to provide his credentials. The client needs to present authentication credentials so that it indeed satisfies the policies associated with the files. The implementation approach for this authentication process is based on the public-key infrastructure (PKI). However, this client-based authentication requires the key manager to have accesses to the association of every client and its satisfied policies. the scalability and flexibility will be limited, if we scale up the number of supported clients and their associations with policies.

To resolve the scalability issue, attribute-based encryption (ABE) [5] [7] [12] [13], turns out to be the most appropriate solution. In particular, our approach is based on Cipher text-Policy Attribute-Based Encryption (CP-ABE) [5]. We summarize the essential ideas of ABE that are sufficient for our FADE design, while we refer readers to for details. Each client first obtains, from the key issuing authority of the ABE system, client is satisfied when an ABE-based private access key that corresponds to a set of attributes. By issuing key authority the client going to present authentication credentials can be done, but we emphasize that this authentication is only a one-time bootstrap process. Later, when a client requests to decrypt the data key of a file on the cloud, the ABE-based public access key that corresponds to the combination of policies associated with the file. It can use its ABE based private access key to recover the data key when the policy combination is satisfied.

FADE uses two independent keys for each policy. The first one is the secret key that is maintained by the key manager for assured deletion. If the secret key is removed from the key manager, files associated with the corresponding policy cannot be recovered. Another one is the ABE-based access key that is used for access control. Corresponding policy is satisfied when the ABE based private key is distributed to the clients, 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. FADE will be enabled by making use of two sets of keys for the same policy to achieve both access control and assured deletion

**4.4 Multiple Key Managers**

We point out that the use of a single key manager will lead to the single-point-of-failure problem. Before the client requests to revoke them, or fail to remove the keys when it is requested to, an untrustworthy key manager may either removes the keys, getting its data back the former case is going to prevent the client, while the latter case may subvert assured deletion. The robustness of the key management service will be improved to minimize its chance of being compromised. Here, we apply Shamir's (M, N) threshold secret sharing scheme [9], where $M \leq N$. Using Shamir's scheme, we divide a secret into N shares and distribute them over N independent key managers, correct shares should 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 $j^{th}$ key manager (where $1 \leq j \leq N$) will independently generate and maintain an RSA public/private control key pair corresponding to a modulus $n_{ij}$. We point out that this key pair is independent of the key pairs generated by other key managers, the same policy $P_i$ will be corresponded by all key pairs. Also, each key manager keeps its own key pair and will not release it to other key managers.

## V. EVALUATIONS

We now evaluate the empirical performance of our implemented prototype of FADE over cloud-analyst tool. There will be increase in data management costs if it is crucial that FADE does not introduce substantial performance or monetary overhead. In addition, the cryptographic operations of FADE should only bring insignificant computational overhead. The FADE will be evaluated on a per-file basis, that is, when it operates on an individual file of different sizes.

 

**5.1 Overall Response Time of FADE**

Here we calculate the response time of our system by considering the factors like encrypting file with different layers of protection and distributing the key to different key managers. So following table: 1 gives the time taken to perform the factors which we considered.

| | Avg (ms) | Min (ms) | Max (ms) |
|---|---|---|---|
| Overall response time: | 234.50 | 160.12 | 364.64 |
| Data Center processing time: | 0.19 | 0.02 | 0.64 |

**Table 1: Overall Response Time of FADE**

**5.2 User Base Hourly Response Time**

Here the below graph shows the response time of user base while integrating with different data centers for uploading/downloading the file.

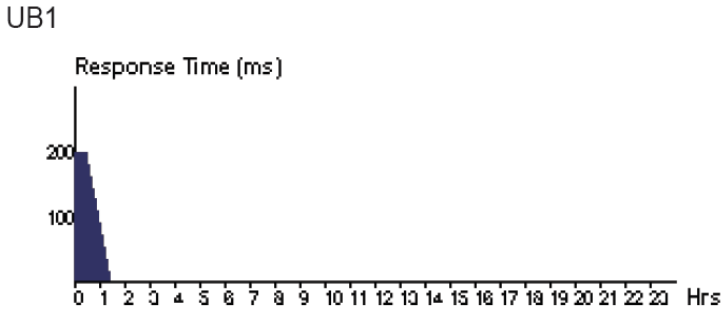


**Figure 2: User Base Hourly Response Time**

**5.3 Data Centre Hourly Loading Time**

Here the below graph shows the response time of data center while integrating with different user bases in order to provide response for different requests of user bases.

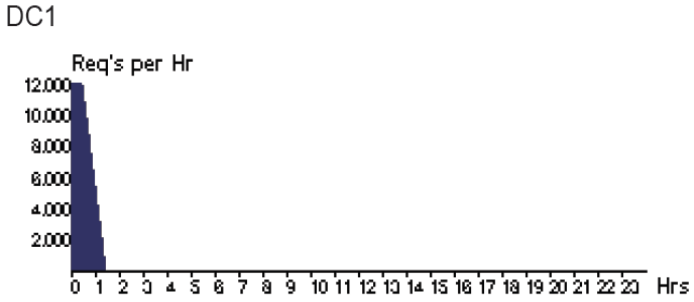


**Figure 3: Data Center Hourly Loading Time**

**5.4 Cost**

Here below table gives the overall cost of the system as per the number of request/response operations performed with respect to a file.

| Data Center | VM Cost $ | Data Transfer Cost $ | Total $ |
|---|---|---|---|
| DC2 | 0.50 | 0.07 | 0.57 |
| DC1 | 0.50 | 0.13 | 0.63 |

**Table 2: Cost**

 

## VI. CONCLUSION AND FUTURE WORK

We propose a cloud storage system called FADE, which aims to provide assured deletion for files that are hosted by today's cloud storage services. Files are assuredly deleted and made unrecoverable by anyone when their associated file access policies are revoked so we are presenting the design of policy based file assured deletion. We present the essential operations on cryptographic keys so as to achieve policy-based file assured deletion. To demonstrate its practicality, and empirically study its performance overhead when it works with windows azure, we implement a prototype of FADE. Our experimental results provide insights into the performance-security trade-off when FADE is deployed in practice. In our future work we can add other security operations and maintaining user record that have all accessed the data.

## REFERENCES

[1] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Luis. A Secure Cloud Backup System with Assured Deletion and Version Control. In 3rd International Workshop on Security in Cloud Computing, 2011.
[2] P. Gutmann. Secure deletion of data from magnetic and solid-state memory. In Proc. of USENIX Security Symposium, 1996.
[3] Amazon. SmugMug Case Study: Amazon Web Services. http://aws.amazon.com/solutions/casestudies/smugmug/, 2006.
[4] Amazon case studies http://aws.amazon.com/solutions/casestudies/#
[5] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute Based Data Sharing with Attribute Revocation. In Proc. of ACM ASIACCS, Apr 2010.
[6] Dropbox. http://www.dropbox.com, 2010.
[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In Proc. of ACM CCS, 2006.
[8] Z. N. J. Peterson, R. Burns, J. Herring, A. Stubblefield, and A. D. Rubin.Secure Deletion for a Versioning File System. In Proc. of USENIX FAST, 2005.
[9] Shamir. How to Share a Secret. CACM, 22(11):612–613, Nov 1979.
[10] R. Perlman. File System Design with Assured Delete. In ISOC NDSS, 2007
[11] R. Perlman, C. Kaufman, and R. Perlner. Privacy-Preserving DRM. In IDtrust, 2010.
[12] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Attribute-Based Systems. In Proc. of ACM CCS, 2006.
[13] Sahai and B. Waters. Fuzzy Identity-Based Encryption. In EUROCRYPT, 2005
[14] B. Schneier. File Deletion. http://www.schneier.com/blog/archives/2009/09/file deletion.html, Sep 2009.
[15] SmugMug.http://www.smugmug.com/, 2010.
[16] W. Stallings. Cryptography and Network Security. Prentice Hall, 2006
[17] M. Vrable, S. Savage, and G. M. Voelker. Cumulus: Filesystem backup to the cloud. ACM Trans. on Storage, 5(4), Dec 2009.

 