ISSN:0975-9646

Secure Cloud Storage with File Assured Deletion

Gousiya Begum , M. Shravya
CSE Department, MGIT, Hyderabad, INDIA,

Abstract — **When we outsource data backup to third-party cloud storage services so as to reduce data management costs, security concerns arise in terms of ensuring the privacy and integrity of out-sourced data. We tried to solve this issue by designing FADE(File Assured Deletion), a practical, implementable and readily deployable cloud storage system that focuses on protecting deleted data with policy-based file assured deletion.**

FADE is built upon standard cryptographic techniques, such that it encrypts outsourced data files to guarantee their privacy and integrity, and most importantly, assuredly deletes files to make them unrecoverable to anyone (including those who manage the cloud storage)upon revocations of file access policies. In particular, the design of FADE is geared toward the objective that it acts as an overlay system that works seamlessly atop today's cloud storage services.

To demonstrate this objective, we implement a working prototype of FADE atop Amazon S3, one of today 's cloud storage services, and empirically show that FADE provides policy-based file assured deletion with a minimal trade-of performance overhead. Our work provide sin sights of how to incorporate value-added security features into current data outsourcing applications.

Keywords: Policy-based file assured deletion, cloud storage, prototype implementation.

I. INTRODUCTION

Cloud storage offers an abstraction of infinite storage space for clients to host data, in a pay-as-you-go manner[1] [3]. For example, Smug Mug [4], a photo sharing website ,chose to host terabytes of photos on Amazon S3 in 2006 and saved about 500K US dollars on storage devices [2].Thus, instead of self-maintaining data centers, enterprises can now outsource the storage of a bulk amount of digitized content to those third-party cloud storage providers so as to save the financial overhead in data management. Apart from enterprises, individuals can also benefit from cloud storage as a result of the advent of mobile devices Given that mobile devices have limited storage space in general, individuals can move audio/video files to the cloud and make active use of space in their mobile devices.

However, privacy and integrity concerns become relevant as we now count on third parties to host possibly sensitive data. To protect outsourced data, a straightforward approach is to apply cryptographic encryption onto sensitive data with a set of encryption keys, yet maintaining and protecting such encryption keys will create another security issue. One specific issue is that upon requests of deletion of cloud storage providers may not completely remove all file copies (e.g., cloud storage providers may make multiple file backup copies and distribute them over the cloud for reliability, and clients do not know the number or even the existence of these backup copies), and eventually have the data disclosed if the encryption keys are unexpectedly obtained either by accidents or by malicious attacks. Therefore, we seek to achieve a major security goal called file assured deletion, meaning that files are reliably deleted and remain permanently unrecoverable and inaccessible by any party.

The security concerns motivate us, as cloud clients, to develop a secure cloud storage system that provides file assured deletion. However, a key challenge of building such a system is that cloud storage infrastructures are externally owned and managed by third-party cloud providers, and hence the system should never assume any structural changes (in protocol or hardware levels) in cloud infrastructures. Thus, it is important to design a secure overlay cloud storage system that can work seamlessly atop existing cloud storage services.

In this paper, we present FADE, a secure overlay cloud storage system that ensures file assured deletion and works seamlessly atop today's cloud storage services. FADE decouples the management of encrypted data and encryption keys, such that encrypted data remains on third-party(untrusted) cloud storage providers, while encryption keys are independently maintained by a key manager service, whose trustworthiness can be enforced using a quorum scheme[13].

A motivating application of FADE is cloud back-up systems (e.g., JungleDisk[5], Cumulus[6]), which use the cloud as the backup storage for files. FADE can be viewed as a value-added security service that further enhances the security properties of the existing cloud-based backup systems.

In summary, our paper makes the following contributions:

We propose a new policy-based file assured deletion scheme that reliably deletes files with regard to revoked file access policies. In this context, we design the key management schemes for various file manipulation operations.

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 .We emphasize that we do not require any protocol and implementation changes on the storage cloud to support our system .Even a naive storage service that merely provides file upload/download operations will be suitable.

We implement a working prototype of FADE using java technologies in NetBeans and we use the OpenSSL library for the cryptographic operations. In addition, we use Amazon S3 as our storage cloud. This section is to address the implementation issues of our FADE architecture based on our experience in prototyping FADE. Our goal is to show the practicality of FADE when it is deployed with today's cloud storage services.

II. Related Work

We present policy-based file assured deletion, the major design building block of our FADE architecture. Our main focus is to deal with the cryptographic key operations that enable file assured deletion. We first review time-based file assured deletion. We then explain how it can be extended to policy-based file assured deletion.

Time-based file assured deletion, which is first introduced in [14], means that files can be securely deleted and remain permanently inaccessible after a predefined duration. The main idea is that a file is encrypted with a data key, and this key is further encrypted with a control key that is maintained by a separated key manager service. Time-based file assured deletion is later prototyped in Vanish [7]. Vanish divides a data key into multiple key shares, which are then stored in different nodes of a peer-to-peer network. Nodes remove the key shares that reside in their caches for 8 hours. If a file needs to remain accessible after 8 hours, then the file owner needs to update the key shares in node caches.

Policy-based Deletion: We associate each file with a single atomic file access policy (or policy for short), or more generally, a Boolean combination of atomic policies. Each (atomic) policy is associated with a control key, and all the control keys are maintained by the key manager. Similar to time-based deletion, the file content is encrypted with a data key, and the data key is further encrypted with the control keys corresponding to the policy combination. When a policy is revoked, the corresponding control key will be removed from the key manager. Thus, when the policy combination associated with a file is revoked and no longer holds, the data key and hence the encrypted content of the file cannot be recovered with the control keys of the policies. In this case, we say the file is deleted. The main idea of policy-based deletion is to delete files that are associated with revoked policies. is associated with a control key, and all the control keys are maintained by the key manager.

We review other related work on protecting outsourced data storage.

Cryptographic protection on outsourced data storage has been considered (see survey in [8]). For example, Wang et al. [9] propose secure outsourced data access mechanisms that support changes in user access rights and outsourced data. Ateniese et al. [10] and Wang et al. [11] propose an auditing system that verifies the integrity of outsourced data. However, all the above systems require new protocol support on the cloud infrastructure , and such additional functionalities may make deployment more challenging.

Security solutions that are compatible with existing public cloud storage services have been proposed. Yun et al.[12] propose a cryptographic file system that provides privacy and integrity gurantees for outsourced data using a universal hash based MAC tree. Goyal et al. [15] extend the idea to key-policy ABE, in which attributes are associated with private keys, and encrypted data can be decrypted only when a threshold of attributes are satisfied.

III. System Design

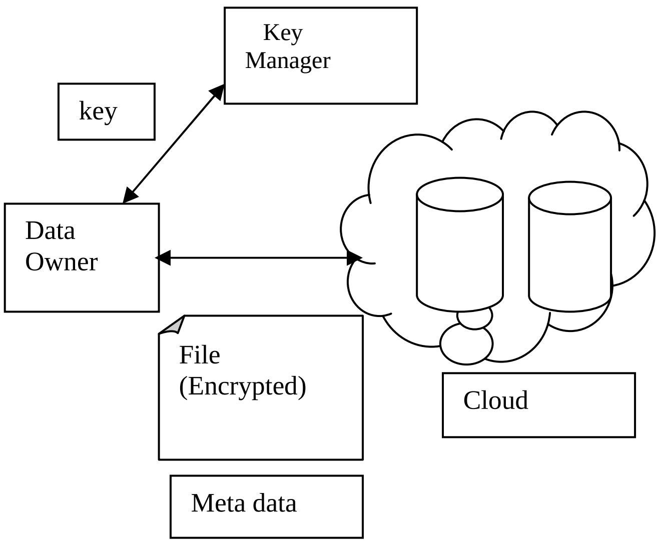


Fig. 1 FADE Architecture

Secure Overlay Cloud Storage with File Assured Deletion is a user-interactive app on the java platform. The project is its own entity and is derived based on the java.

We design FADE, a practical implementable, and readily deployable cloud storage system that focuses on protecting deleted data with policy-based file assured deletion. FADE is built upon standard cryptographic techniques ,such that it encrypts outsourced data files to guarantee their privacy and integrity ,and most importantly, assuredly deletes files to make them unrecoverable to anyone (including those who manage the cloud storage)upon revocations of file access policies.

The application uses many different software interfaces. The Windows XP operating system is essential for the product to operate. Net Beans will be needed for the development portion of the project, and it will be utilizing the java software packages. Communication is not needed by this application because it is based locally.

The physical characteristics of the application consist of various java app that run the java environment

There is no need of communication hardware in Project FADE: Secure Overlay Cloud Storage with File Assured Deletion.

We define the metadata of FADE attached to individual files and also describe how we implement the data owner and the key manager, and how the data owner interacts with the storage cloud.

Representation of Metadata:

For each file protected by FADE, we include the metadata that describes the policies associated with the file as well as a set of encrypted keys. In FADE, there are two types of metadata:

- File metadata: The file metadata mainly contains two pieces of information file size and hash. We hash the encrypted file and it is attached at the beginning. Both the file metadata and the encrypted data file will then be treated as a single file to be uploaded to the storage cloud.

- Policy metadata: The policy metadata includes the specification of the Boolean combination of policies and the corresponding encrypted cryptographic keys. Here, we assume that each single policy is specified by a unique 4-byte integer identifier. To represent a Boolean combination of policies, we express it in disjunctive canonical form, i.e., the disjunction (OR) of conjunctive policies, and use the characters '*' and '+' to denote the AND and OR operators. Then we upload the policy metadata as a separate file to the storage cloud. This enables us to renew policies directly on the policy metadata without retrieving the entire file from the storage cloud. In our implementation, individual files have their own policy metadata, although we allow multiple files to be associated with the same policy. In other words, for two data files that are under the same policy, they will have different policy metadata files that specify different data keys, and the data keys are protected by the control key of the same policy.

Data Owner and Storage Cloud:

Function calls:

- Upload(file, policy) : The data owner encrypts the input file using the specified policy (or a Boolean combination of policies). It then sends the encrypted file and the metadata onto the cloud. In our implementation, the file is encrypted using the RSA algorithm, yet we can adopt a different symmetric-key encryption algorithm depending on applications.
- Download(file) : The data owner retrieves the file and the policy metadata from the cloud, checks the integrity of the file, and decrypts the file.
- Delete(policy) : The data owner tells the key manager to permanently revoke the specified policy.All files associated with the policy will be assuredly deleted.
- Renew(file, new_policy) : The data owner first fetches the policy metadata for the given file from the cloud. It then updates the policy metadata with the new policy. Finally, it sends the policy metadata back to the cloud.

The above function calls can be exported as library APIs that can be embedded into different implementations of the data owner. In our current prototype, we implement the data owner as a user-level program that can access files under a working directory of a desktop PC. The above exported interfaces wrap the third-party APIs for interacting with the storage cloud.

Key Manager:

Function calls:

- Creating a policy. The key manager creates a new policy and returns the corresponding public control key.
- Retrieving the public control key of a policy. If the policy is accessible, then the key manager returns the public control key. Otherwise, it returns an error.
- Decrypting a key with respect to a policy. If the policy is accessible, then the key manager decrypts the (blinded) key. Otherwise, it returns an error.
- Revoking a policy. The key manager revokes the policy and removes the corresponding keys.

We implement the basic functionalities of the key manager so that it can perform the required operations on the cryptographic keys. In particular, all the policy control keys are built upon 1024-bit blinded RSA.

Constraints: Typical Project Constraints

Resource Constaints

- Computer resources will be available on a limited basis.
- Key customer resources will be available on a restricted basis.

Environmental Constraints

- The development or operating environment is new, and no project new members are familiar with it.
- K**ey** decision-makers are difficult to contact when issues arise.
- The project environment is new and the components have not yet been successfully integrated.
- The project depends upon the successful and timely completion of associated projects.

Budgetary Constraints

- Statistics used in preparing the estimates are unreliable.
- Outside consulting requirements cannot be accurately estimated.

Functionality Constraints

• The scope of the project is unclear.
• The project depends upon receiving data from other, external applications.

IV. IMPLEMENTATION

Our design is based on blinded RSA in which 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 . For each policy i, the key manager generates two secret large RSA prime numbers pi and qi and computes the product ni = piqi . The key manager then randomly chooses the RSA public-private control key pair (ei, di). The parameters (ni, ei) will be publicized, while di is securely stored in the key manager. On the other hand, when the data owner encrypts a file F, it randomly generates a data key K, and a secret key Si that corresponds to policy Pi. We let {m}k denote a message m encrypted with key k using asymmetric-key encryption (e.g., RSA). We let R be the blinded component when we use blinded RSA for the exchanges of cryptographic keys. Suppose that F is associated with policy Pi. Our goal here is to ensure that K, and hence F, are accessible only when policy Pi is satisfied.

File upload: We set the private key value and also the expiry date for the file and choose the file to be uploaded and then uploading is completed and we get a random generated public control key. The file that is uploaded is in encrypted format by the RSA public key and is stored in the folder in the cloud with the user folder name who is uploading.

File download: When we wish to download a file from the data files we uploaded we need to first generate the control key for it that is done by entering the privacy key value we set along with the public control key that got generated when file was uploaded this is result in giving us the public and private keys by the RSA algorithm. The file can be downloaded by entering the private control key that got generated. The downloaded file gets stored in the folder downloads in the cloud in decrypted format.

Request: The data owner can request the admin to give the details about the keys if incase he forgets them. The keys are sent to the mail id of the data owner in this way the keys are secure.

Delete File: The data owner tells the key manager to permanently revoke the specified policy. All files associated with the policy will be assuredly deleted by entering the password.

Policy Revocation for File Assured Deletion: If a policy Pi is revoked, then the key manager completely removes the private key di and the secret prime numbers pi and qi. Thus, we cannot recover Si from Siei , and hence cannot recover K and the file F . We say that the file F , which is tied to policy Pi, is assuredly deleted. Note that the policy revocation operations do not involve interactions with the storage cloud.

Multiple Policies: In addition to one policy per file, FADE supports a Boolean combination of multiple policies. We mainly focus on two kinds of logical connectives: (i) the conjunction (AND), which means the data is accessible only when every policy is satisfied; and (ii) the disjunction (OR), which means if any policy is satisfied, then the data is accessible.

Conjunctive Policies: Suppose that F is associated with conjunctive policies P1 $\wedge$ P2 $\wedge \cdot\cdot\cdot \wedge$ Pm. To upload F to the storage cloud, the data owner first randomly generates a data key K, and secret keys S1, S2 . , Sm. It then sends the following to the storage cloud: {{K}S1}S2 · · ·Sm, Se1 , Se2 ., Sem m , and {F}K. On the other hand, to recover F, the data owner generates a random number R and sends (S1R)e1 , (S2R)e2 , ..., (SmR)em to the key manager, which then returns S1R, S2R, . . . , SmR. The data owner can then recover S1, S2, . . . , Sm, and hence K and F.

Disjunctive Policies: Suppose that F is associated with disjunctive policies Pi1 $\vee$ Pi2 $\vee \cdot\cdot\cdot \vee$ Pim. To upload F to the cloud, the data owner will send the following: {K}S1 , {K}S2 , . . ., {K}Sm, Se1 , Se2 , . . ., Sem m , and {F}K. Therefore, the data owner needs to compute m different encrypted copies of K. On the other hand, to recover F, we can use any one of the policies to decrypt the file, as in the above operations.

Policy Renewal: We conclude this section with the discussion of policy renewal. Policy renewal means to associate a file with a new policy (or combination of policies). For example, if a user wants to extend the expiration time of a file, then the user can update the old policy that specifies an earlier expiration time to the new policy that specifies a later expiration time.

V. EXPERMENTAL RESULTS AND ANALYSIS

Home Page : When we click on the home page option it displays the details about the project.

Login Page : In this option page we need to enter the Username, Password and click on the login link. If administrator enters ID and password correct it goes to the admin services otherwise displays the same page with an error message. If user enters ID and password correct it goes to the user services otherwise displays the same page with an error message.

Register Page : In this page we click on register and provide the corresponding details with this the registration is done successfully.

Administrator Module :

Login : In this page enter the username, password and click login. If administrator enters ID and password correct it goes to the admin services otherwise displays the same page with an error message.

Admin Profile : In this when we click on view request link. The admin will see all types of user's data.

User Details : When we Click on user details link. The selected user details will be retrieved and displayed.

Delete User : When we click on delete user link. The entered user name details checks in database and is deleted successfully.

Cloud Details : By clicking on cloud link the selected category details are retrieved and displayed.

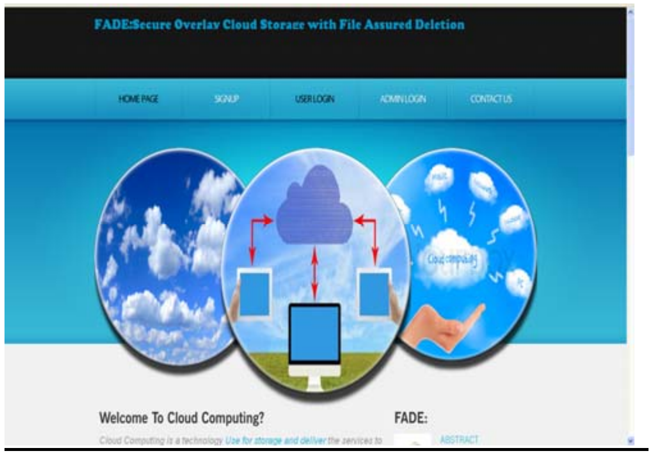
User Module :

Login : When we enter the username, password and click on the login link.If Employee enters ID and password correct it goes to the other page otherwise displays the same page with an error message to be viewed by the user.

Upload File : In this we browse a file and click on upload button and then the selected file gets stored in required area and displays message upload successful.

Data Files : When we click on request . As per the login details or else the information provided by the user file details will be generated and then the retrieved details will be displayed.

Delete File : When we click on delete link. As per the user login details the selected file will be deleted from the user account and the deleted file will not be appear in the file list.


FADE:Secure Overlay Cloud Storage with File Assured Deletion
HOME PAGE
SIGNUP
USER LOGIN
ADMIN LOGIN
CONTACT US
Cloud computing
Welcome To Cloud Computing?
FADE:
ABSTRACT

Cloud Computing is a technology Use for storage and deliver the services to

Fig. 2 Home page

In this page we have the options to go to home age, signup, userlogin, admin login and contact us. By clicking on anyone of these links we will be taken to the respected linked pages.

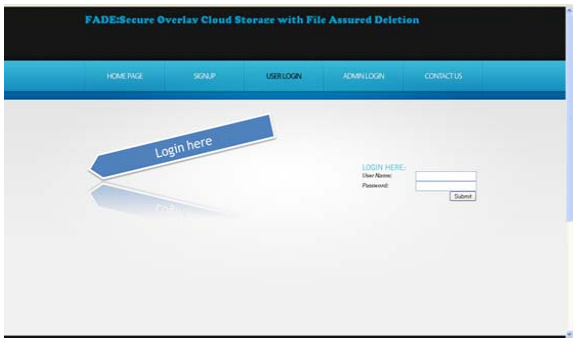
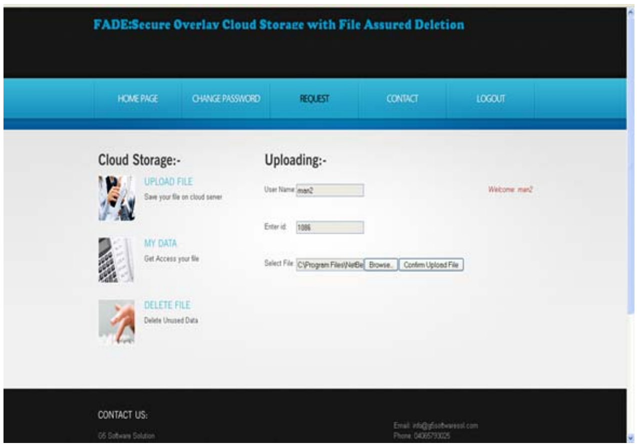


Fig. 3 User/Admin Login

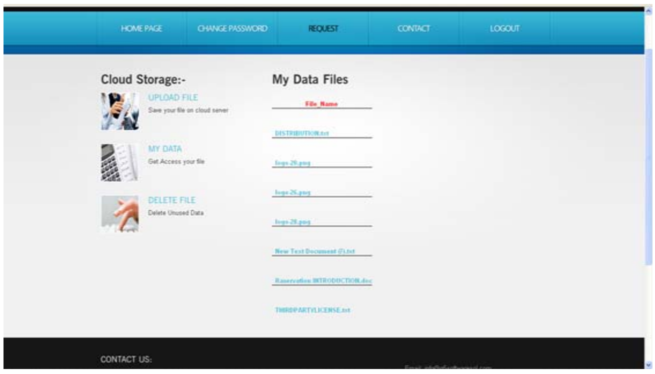
In this page we need to enter the username and password and click login it will check if it is authentic user and then login successfully and open up the user profile with the options of request, change password, change policy, upload file, delete file, view details.



User Name: man2
Enter id: 1086
Select File: C:\Program Files\NetBe

Fig. 4 File Upload

By clicking the upload file link we will be taken to this page with details to be entered about the user name, enter the id, select the policy to be implemented and then select the file to be uploaded and click on confirm uploading file. We will get a page displaying upload successful.



File_Name
DISTRIBUTION.txt
logo-20.png
logo-26.png
logo-28.png
New Text Document (2).txt
Raservation INTRODUCTION.doc
THIRDPARTYLICENSE.txt

Fig. 5 File Download

In this page we can download the file but first we need to first generate a control key and enter the private key and download the required file.

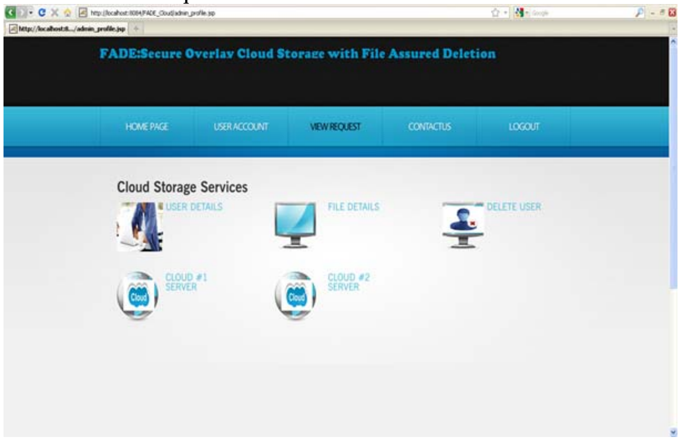
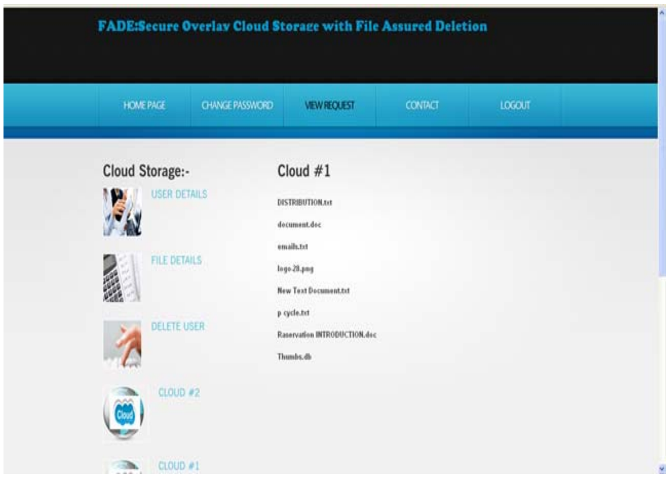


Fig. 6 Admin Page

The admin when he logs in he has the options of viewing the cloud servers, user details, file details, delete user, view request.



DISTRIBUTION.txt
document.doc
emails.txt
logo-28.png
New Text Document.txt
p cycle.txt
Raservation INTRODUCTION.doc
Thumbs.db

Fig. 7 Virtual Machine

When the admin clicks on the cloud server he can view the files that got uploaded into the server.

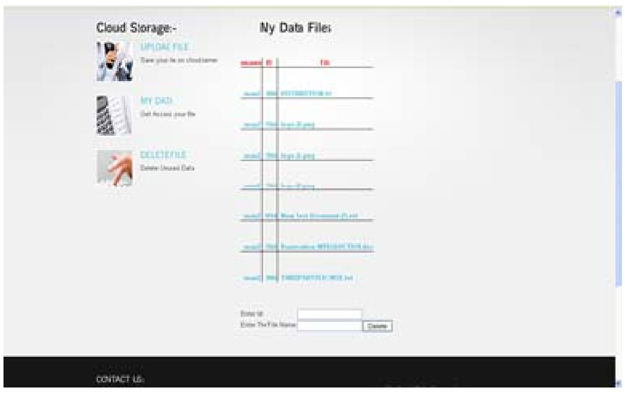


Fig. 8 Delete Files

The user can delete files by entering the file name and the id and click delete button.

VI. CONCLUSION

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 .We present the design of policy-based file assured deletion ,in which files are assuredly deleted and made unrecoverable by anyone when their associated file access policies are revoked .We present the essential operations on cryptographic keys so as to achieve policy-based file assured deletion .We implement a prototype of FADE to demonstrate its practicality, and empirically study its performance over head when it works with Amazon S3.Our experimental results provide insights into the performance-security trade-off when FADE is deployed in practice.

REFERENCES

[1] Yang Tang, Patrick P. C. Lee, John C. S. Lui, and Radia Perlman, FADE: Secure Overlay Cloud Storage with File Assured Deletion, *IEEE Transactions Dependable on secure computing,* VOL:9 NO:6 YEAR 2012.

[2] Amazon. SmugMug Case Study: Amazon Web Services .http*://aws.amazon.com/solutions/case-studies/smugmug/*, 2006.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee,D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley,* Feb 2009.

[4] SmugMug. http://www.smugmug.com/.

[5] JungleDisk. http://www.jungledisk.com/

[6] M. Vrable, S.Savage, and G. M. Voelker. Cumulus: Filesystem backup to the cloud. *ACM Trans. On Storage(ToS),* 5(4), Dec 2009.

[7] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. *In Proc. of USENIX Security Symposium,* Aug 2009.

[8] S. Kamara and K.Lauter. Cryptographic Cloud Storage. *In Proc. Of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization*, 2010.

[9] W. Wang, Z. Li, R. Owens, and B. Bhargava. Secure and Efficient Access to Outsourced Data. *In ACM Cloud Computing Security Workshop (CCSW)*, Nov 2009.

[10] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik. Scalable and Efficient Provable Data Possession. *In Proc. of SecureComm*, 2008.

[11] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving pubic auditing for storage security in cloud computing. In Proc. of IEEE INFOCOM, Mar 2010.

[12] A.Yun, C. Shi and Y.Kim. On Protecting Integrity and Confedentiality of Cryptographic File System for Outsourced Storage. In ACM Cloud Computing Security Workshop (CCSW), Nov 2009.

[13] A. Shamir. How to Share a Secret. CACM, 22(11):612–613, Nov 1979.

[14] R. Pearlman. File System Design with Assured Delete. *In ISOC NDSS*, 2007.

[15] 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.