Cloud Access Security On File System Using Secure Policies For Jelastic Cloud

Priyanka Khandelwal[1], Rakesh Sharma[2]

Dept. Of C.S.E, R.C.E.W,Jaipur

[1]priyankakhandelwal85@yahoo.co.in
[2] mtech@rcew.ac.in

***Abstract* – Now a days we can outsource data backups off-site to third-party cloud storage services(Jelastic cloud) by which we can reduce data management costs. However, we need to provide security guarantees for the outsourced data, maintained by third parties. In this paper we design and implement FADE, a secure overlay cloud storage system which is able to achieve fine-grained, policy-based access control and file assured deletion. It associates the outsourced files with file access policies, and assuredly deletes files to make them unrecoverable by anyone upon revocations of file access policies. For achieving such security goals, FADE is built upon a set of cryptographic key operations that are self-maintained by a quorum of key managers that are independent of third-party clouds. Particularly, FADE acts as an overlay system which works seamlessly atop today's cloud storage services. We implement a proof-of-concept prototype of FADE Jelastic, one of today's cloud storage services. We conduct extensive empirical studies, and demonstrate that FADE provides security protection for outsourced data, while introducing only minimal performance and monetary cost overhead. Our work provides insights of how to incorporate value-added security features into today's cloud storage services.**

***Keywords*—Access control, Assured deletion, backup/ recovery, cloud storage**

I. INTRODUCTION

Now a days *Cloud storage* become a new business solution for remote backup outsourcing, as it offers an abstraction of infinite storage space for clients to host data backups in a pay-as- you-go manner. It helps enterprises and government agencies significantly in reducing their financial overhead of data management, now they can archive their data backups remotely to third-party cloud storage providers rather than maintain data centers on their own..

As we now outsource the storage of sensitive data to third parities,so security concerns become necessary. In our implementation, we are particularly interested in two security issues. First, we need to provide guaran- tees of *access control*, in which we must ensure that only authorized parties can access the outsourced data on the cloud. In particular, we must restrict third-party cloud storage providers from burrowing any sensitive information of their clients' data for their own marketing purposes. Second, it is important to provide guarantees of *assured deletion*, meaning that outsourced data is permanently inaccessible to anybody (including the data owner) upon requests of deletion of data.

In our implementation, we present FADE, a secure overlay cloud storage system that provides fine-grained access control and assured deletion for outsourced data on the Jelastic cloud using secure policies.

In FADE, active data files that live on the cloud are associated with a set of user-defined *file access policies* like., time expiration, read/write permissions of authorized users and these data files are accessible only to users who satisfy the file access policies. The design perception of FADE is to decouple the management of encrypted data and cryptographic keys, such that encrypted data remains on third-party, cloud storage providers, while cryptographic keys are independently maintained and operated by a quorum of key managers that altogether from trustworthiness. To provide guarantees of access control and assured deletion, FADE leverages off-the-shelf cryptographic schemes including threshold secret sharing and attribute based encryption and performs various cryptographic key operations that provide security protection for basic file upload/download operations. We implement a proof- of-concept prototype of FADE to validate its feasibility, and export a set of library APIs that can be used, as a value-added security service, to boost the security properties of general data outsourcing applications.

II. SYSTEM ANALYSIS

A. Existing System

In the existing system Time-based file assured deletion, is introduced, which means that files can be securely deleted and remain permanently inaccessible after a pre-

defined duration. The main idea behind it, that a file is encrypted with a *data key* by the owner of the file, and this data key is again encrypted with a *control key* by a separate key manager. The key manager is a server that is responsible for cryptographic key management. In, the control key is *time-based*, meaning is that it will be completely removed by the key manager as an expiration time is reached, the expiration time is specified at the time the file is first declared. Without the control key, the data key and hence the data file remain encrypted and are deemed to be inaccessible.

1. Problems Of Existing System

- ✓ Without the control key, the data key and hence the data file remain encrypted and are deemed to be inaccessible.
- ✓ The main security property of file assured deletion is that even if a cloud provider does not remove expired file copies from its storage those files remain encrypted and unrecoverable.

B. Proposed System

In our implementation we propose a new Policy based assured deletion scheme called FADE, 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 main idea behind it, that a each file is encrypted with a *File key or File policies* and then each policy is associated with a *control key* and all the *control keys* are maintained by a Key manager for example file is associated with a *file key or file policy* and the file content is encrypted with a *data key* , and this data key is again encrypted with a *control key* corresponding to policy.So when the policy is revoked the corresponding a *control key* is removed from the key manager.Thus when the policy associated with a file is revoked and no longer holds,the data key and hence the encrypted content of the file cannot b recovered with the *control key* of the policy so we can say that file is assuredly deleted.

1. Design of Fade

- ✓ Work atop today's cloud(Jelastic cloud) as an overlay.
- ✓ Achieve protection from cloud client's perspective, no changes on the cloud provider side.

2. Security of Fade

✓ Fine-grained file assured deletion: Files are accessed control and assured deletion.

III. FADE OVERVIEW

In this paper we analyze the design of FADE Jelastic,A system which provides guarantees of access control and assured deletion for the outsourced data in cloud storage. We show the necessary components of FADE, and state the design and security goals that it seeks to achieve.Figure 1 illustrates an overview of the FADE system.The cloud hosts data files on behalf of a group of FADE users who want to outsource data files to the cloud based on their definitions of file access policies. FADE can be seen as an overlay system atop the underlying cloud.It applies security protection to the outsourced data files before they are hosted on the cloud.

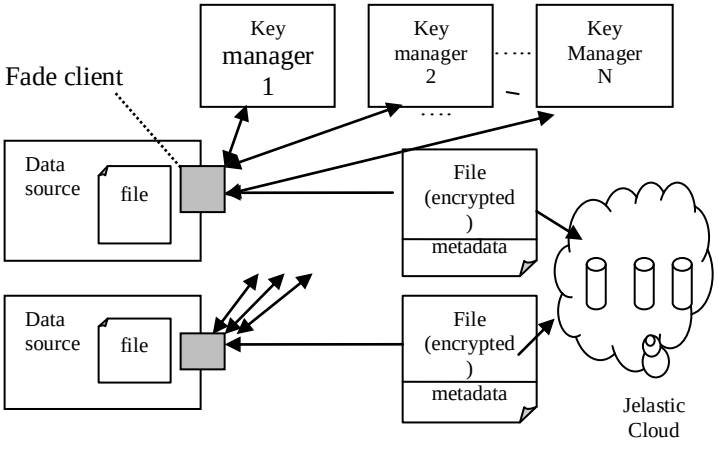

Fig. 1: The FADE system. Each client (deployed locally with its own data source) interacts with one or multiple key managers and uploads/downloads data files to/from the cloud.

A. FADE Entities

As shown in Figure 1, the FADE system is made up of three main entities:

1. FADE clients

A *FADE client* is an interface that bridges the data source or filesystem and the cloud. It applies encryption/decryption to the outsourced data files uploaded to /downloaded from the cloud. It also communicates with the key managers to perform the necessary cryptographic key operations.

2. Key managers

We implement a quorum of key managers, each of which supports two major types of functions:

- ✓ *policy manage- ment*, in which a key manager creates or revokes policies, as well as their associated control keys (for assured deletion) and access keys (for access control),
- ✓ *key management*, in which a key manager performs the encryption or decryption on the (blinded) data key.We implement the basic

functionalities of a key manager so that it can perform the required operations on the cryptographic keys..

3.Jelastic cloud

Jelastic is a plateform-as-a-service(PaaS) cloud computing service that provides network,server and storage solutions to their clients.

B. Deployment

In our current design, a FADE Jelastic client is deployed locally with its corresponding data source as a local driver. It is also possible to deploy the FADE client as a cloud storage proxy , so that it can interconnect multiple data sources. In deployment, we use standard TLS/SSL to protect the communication between each data source . In FADE, the set of key managers is deployed as a centralized trusted service, whose truthfulness is enforced through a quorum scheme. We assume that the key managers are centrally maintained, for example, by the system administrators of an enterprise that deploys FADE . We note that this centralized control is opposed to the core design of Vanish which proposes to use decentralized key management on top of existing P2P DHT systems. However, there is no straightforward solution to develop fine-grained cryptographic key management operations over a decentralized P2P DHT system. In view of this, we propose to deploy a centralized key management service, and use a quorum scheme to improve its robustness.

C . Cryptographic Keys

In our implementation FADE Jelastic defines Five types of cryptographic keys to protect data files stored on the cloud:

1. Personal Key

A personal key is a random secret key that is generated by cloud third party storage providers to the client as the client registers.It is associated with clients policy.It is used for encryption via symmetric-key encryption(AES).In AES 128-bit plain text block is combined with 128-bit key which gives the possible range of 2 raise to power 128 keys.which is a very wide range.

2. Data key

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

3. Control key

A control key is associated with a particular policy. It is represented by a public-private key pair, 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 control key forms the basis of policy-based assured deletion.it is used for encrypting and decrypting files via MD5(message digestive algo.). to generate this key the input text is processed in 512 bit blocks and then the output is a set of four 32 –bit blocks.which make up the 128-bit message digest.MD5 adds much as much as complexity and randomness so that no two message digest produced by MD5 on any two different messages are equal.

4 . Access key.

Similar to the control key, an access key is associated with a particular policy, and is represented by a public-private key pair. However, unlike the control key, the private access key is maintained by a FADE client that is authorized to access files of the associated policy. The access key is built on attribute-based encryption , 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. Without any of these keys, it is computationally infeasible to recover an outsourced file being protected by FADE. It is used for encryption decryption via RSA algorithm.RSA algo is based on the mathematical fact in it the public private key pair is based on the very large prime numbers.

5.File Key

A File key is a secret key which is assigned to the data files. it is used every time client deletes,updates,revokes the file or policies.File key is associated with file policies , each time when data owner/client revokes/renew the policies associated with file it is needed to satisfied.the encryption or decryption of file key takes place using RSA algorithm.

IV. IMPLEMENTATION

We implement a working prototype of FADE Jelastic using JAVA on Windows95/98/2000/XP .Our implementation is built on off-the-shelf library APIs. Specifically, we use the OpenSSL library for the cryptographic operations, the cpabe library for the ABS-based access control, and the ssss library for sharing control keys to a quorum of key managers. The ssss library is originally designed as a command-line utility to deal with keys in ASCII format. We slightly modify ssss and add two functions to split and combine keys in binary format, an so as to make it compatible with other libraries. we use Jelastic (Platform-as-a-service)clouds as our cloud storage backend because Jelastic solutions provide the maximum application density, the fastest

deployment model and the easiest management for private, public and hybrid clouds, all while retaining the flexibility to customize infrastructure and application configurations.Also in our implementation we use JMS(Java messaging service) to send messages to users.
In the following, we define the metadata of FADE being attached to individual data files. We then describe the modules of the system and the basic operationsthat take place between clients,kemanager and cloud storage.

A. System Module

In our implementation ,After analyzing carefully system should consist of following modules

1. Data Owner/Client 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.

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. 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.StorageCloud(Third party provider) Module

The storage cloud is maintained by a third-party cloud provider 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.

4.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 Seii , and hence cannot recover K and the file F. We say that the file F, which is tied to policy Pi, is assuredly deleted.

B . Representation of Metadata

For every data file protected by FADE, we include the metadata. Metadata describes the policies that are associated with the file as well as a set of cryptographic keys. More precisely, the metadata contains the specification of the Boolean combination of policies, and the corresponding crypto- graphic keys including the encrypted File key, data key of the file and the control keys associated with the policies. Here, we assume that each (atomic) policy is specified by a unique 4-byte integer identifier. To represent a Boolean combination of policies, we express it in *disjunctive canon- ical form*, i.e., the disjunction (OR) of conjunctive policies, and use the characters '∗' and '+' to denote the AND and OR operators. We upload the metadata as a separate file to the cloud. This enables us to renew policies directly on the metadata file without retrieving the entire data file from the cloud.In our implementation, individual data files have their own metadata, each specifying its own data key.

C. Basic operations of FADE:

Our implementation uses four function calls to enable end users to interact with the cloud:

1. Upload(file, policy)

The client encrypts the input file according to the specified policy (or a Boolean combination of policies). Here, the file is encrypted using the 128-bit AES algorithm with the cipher block chaining (CBC) mode. After en- cryption, the client also appends the encrypted file size (8 bytes long) and the HMAC-SHA1 signature (20 bytes long) to the end of encrypted file for integrity checking in later downloads. It then sends the encrypted file and the metadata onto the cloud.

2. Download(file)

The client retrieves the file and policy metadata from the cloud. It then checks the integrity of the encrypted file, and decrypts the file.

3.Revoke(policy)

The client tells the key managers to permanently revoke the specified policy. All files associated with the policy will be assuredly deleted. If a file is associated with the conjunctive policy combination that contains the revoked policy, then it will be assuredly deleted as well.

4.Renew(file, new_policy)

The client first needs to provide the *file key* if it satisfies than,fetches the metadata of the given file from the cloud. It updates the metadata with the new

policy. Finally, it sends the metadata back to the cloud. Note that the operation does not involve transfer of the input file.We export the above function calls exported as library APIs. Thus, different implementations of the client can call the library APIs and have the protection offered by FADE. In our current prototype, we implement the client as a user-level program that can access files under a specified folder.

V. CONCLUSION

In this paper we propose a Practical Cloud Storage system Called FADE,which aims to provide access control assured deletion for files that are hosted by today's cloud storage services(third party storage provider)using secure policies.We associate files with File keys or File policies that controls how files can be accessed.We describe the essential operations on cryptographic so as to achieve access control and assured deletion,FADE also leverages existing cryptographic techniques..we implement a prototype of FADE to demonstrate its practicality and empirically study its performance overhead when it works with Jelastic Servlent.Our experimental results provide insights into the performance security trade-off when FADE is deployed in practice.
Future work:It can be implemented on multiple clouds.

REFRENCES

[1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. RACS: A Case for Cloud Storage Diversity. In Proc. of ACM SoCC, 2010.
[2] S. Kamara and K. Lauter.Cryptographic Cloud Storage. In Proc. of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization, 2010.
[3] R. Perlman, C. Kaufman, and R. Perlner.Privacy-Preserving DRM.InIDtrust, 2010.[25] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Attribute-Based Systems. In Proc. of ACM CCS, 2006.
[4] Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui. A Secure Cloud Backup System with Assured Deletion and Version Control.In 3rd International Workshop on Security in Cloud Computing, 2011.
[5] W. Stallings. Cryptography and Network Security.Prentice Hall, 2006.
[6] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman. FADE: Secure Overlay Cloud Storage with File Assured Deletion. In Proc.Of ICST SecureComm, 2010.
[7] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for storage security in cloud computing. In Proc. of IEEE INFOCOM, Mar 2010.
[8] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman. FADE: Secure Overlay Cloud Storage with File Assured deletion.