E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

DYNAMIC ACCESS CONTROL AND FILE ASSURED DELETION FOR SECURED CLOUD STORAGE

Mrs. Priyanka Nagtilak, Prof. Archana Lomte
Department of Computer Engineering
Bhivarabai Sawant Institute of Technology & Research (W)
Wagholi, Pune, Maharashtra, India.411027
Email- priyankaburgute10@gmail.com
archanalomte@gmail.com

ABSTARCT:

This paper describes a system that supports high availability of data, until the data should be expunged, at which time it is impossible to recover the data. This design supports assured deletion of files. As we can now 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 outsourced data. We design dynamic file assured deletion; a practical, implementable, and readily deployable cloud storage system that focuses on protecting deleted data with policy-based file assured deletion. DFADE 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 DFADE is geared toward the objective that it acts as an overlay system that works seamlessly atop today's cloud storage services. This paper provides insights of how to incorporate value-added security features into current data outsourcing applications.

Keywords: *file; cloud storage; data services; prototype implementation.*

1. INTRODUCTION

In this paper, we present DFADE, a secure overlay cloud storage system that ensures dynamic file assured deletion and works seamlessly atop today's cloud storage services. DFADE 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 [18]. DFADE generalizes time-based file assured deletion [5, 14] (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 more flexible file 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.

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. We implement a working prototype of DFADE atop Amazon S3 [2]. Our implementation aims to illustrate that various applications can benefit from DFADE, such as cloud-based backup systems. DFADE consists of a set of API interfaces that we can export, so that we can adapt DFADE into different cloud storage implementations. We empirically evaluate the performance overhead of DFADE atop Amazon S3, and using realistic experiments, we show the feasibility of DFADE in improving the security protection of data storage on the cloud.

Cloud storage (e.g., Amazon S3 [2], MyAsiaCloud [11]) offers an abstraction of infinite storage space for clients to host data, in a pay-as-you-go manner [3]. 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 (e.g., smartphones, laptops). Given that mobile devices have limited storage space in general, individuals can move audio/video files to the cloud and make effective 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 files, 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.

To achieve security goals, file assured deletion 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. We address the problem of resource management for a large-scale cloud environment that hosts sites. Our contribution centers on outlining distributed middleware architecture and presenting one of its key elements, a gossip protocol that meets our design goals: fairness of resource allocation with respect to hosted sites, efficient adaptation to load changes and scalability in terms of both the number of machines and sites. We formalize the resource allocation problem as that of dynamically maximizing the cloud utility under CPU and memory constraints. While we can show that an optimal solution without considering memory constraints is straightforward but not useful, we provide an efficient heuristic solution for the complete problem instead.

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.

We evaluate the protocol through simulation and find its performance to be well aligned with our design goals. While we can now 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 outsourced data. We design DFADE; a practical, implementable, and readily deployable cloud storage system that focuses on protecting deleted data with policy-based file assured deletion. DFADE 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 DFADE is geared toward the objective that it acts as an overlay system that works seamlessly atop Amazon S3, one of today's cloud storage services, and empirically show that DFADE provides policy-based file assured deletion with a minimal trade-off of performance overhead. Our work provides insights of how to incorporate value-added security features into current data outsourcing applications.

2. CLOUD COMPUTING

Cloud computing is the long dreamed vision of computing as a utility, where cloud customers can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources [2]. The benefits brought by this new computing model include but are not limited to: relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc. [3]. As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud, such as emails, personal health records, company finance data, and government documents, etc. The fact that data

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

owners and cloud server are no longer in the same trusted domain may put the outsourced unencrypted data at risk [4]: the cloud server may leak data information to unauthorized entities [5] or even be hacked [6].

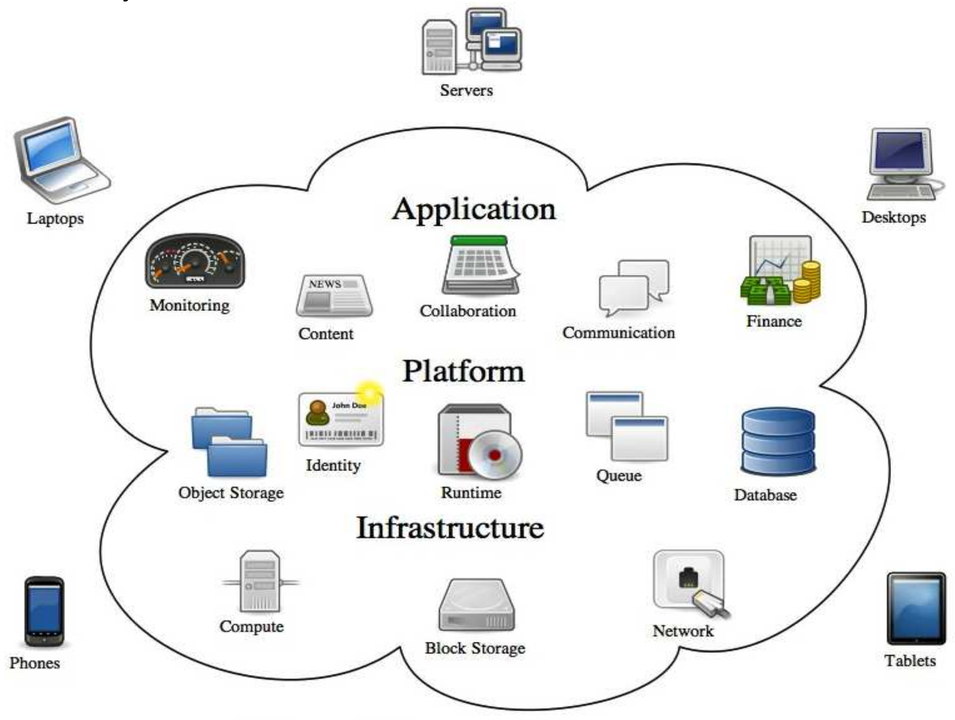


Figure 2.1: Cloud Computing

3. POLICY BASED DELETION

We now generalize time-based deletion to policy-based deletion as follows. 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. Suppose now that a file is associated with a single policy. Then similar to time-based deletion, the file content is encrypted with a data key, and the data key is further 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. 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 be recovered with the control key of the policy. In this case, we say the file is assuredly deleted. The main idea of policy-based deletion is to delete files that are associated with revoked policies.

The definition of a policy varies across applications. In fact, time-based deletion is a special case under our framework. In general, policies with other access rights can be defined. To motivate the use of policy-based deletion, let us consider a scenario where a company outsources its data to the cloud. We consider four practical cases where policy-based deletion will be useful.

3.1 Storing files for tenured employees: For each employee (e.g., Alice), we can define a user-based policy “P: Alice is an employee”, and associate this policy with all files of Alice. If Alice quits her job, then the key manager will expunge the control key of policy P. Thus, nobody including Alice can access the files associated with P on the cloud, and those files are said to be deleted.

3.2 Storing files for contract-based employees: An employee may be affiliated with the company for only a fixed length of time. Then we can form a combination of the user-based and time-based policies for employees’ files. For example, for a contract-based employee Bob whose contract expires on 2010-01-01, we have two policies “P1: Bob is an employee” and “P2: valid before 2010- 01-01”. Then all files of Bob are associated with the policy combination P1 ^ P2. If either P1 or P2 is revoked, then Bob’s files are deleted.

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

3.3 Storing files for a team of employees: The company may have different teams, each of which has more than one employee. As in above, we can assign each employee i a policy combination Pi1 ^ Pi2, where Pi1 and Pi2 denote the user-based and time-based policies, respectively. We then associate the team's files with the disjunctive combination (P11 ^ P12) V (P21 ^ P22) V · · · V (PN1^ PN2) for employees 1, 2. . . N. Thus, the team's files can be accessed by any one of the employees, and will be deleted when the policies of all employees of the team are revoked.

4. OVERVIEW OF DFADE SYSTEM

We now overview the design of DFADE, a system that provides guarantees of access control and assured deletion for outsourced data in cloud storage. We present the necessary components of DFADE, and state the design and security goals that it seeks to achieve.

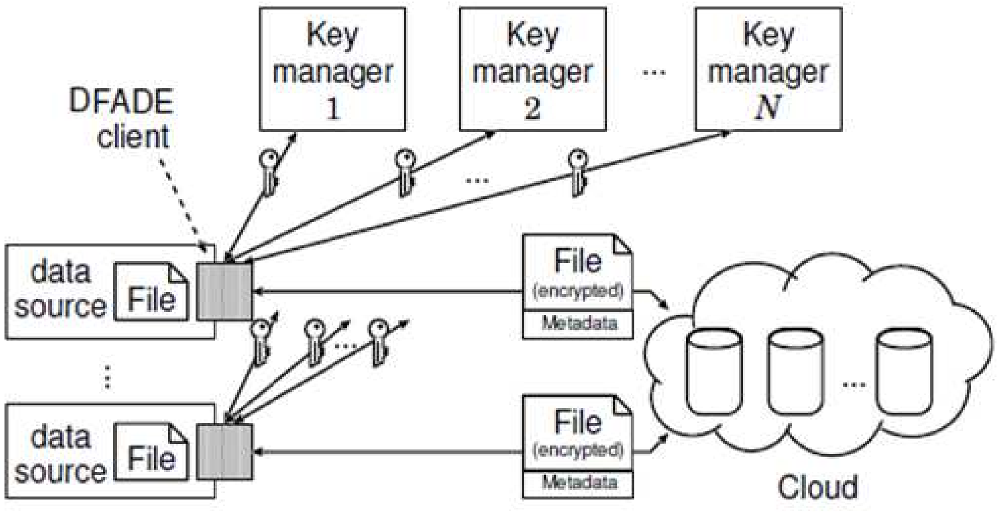
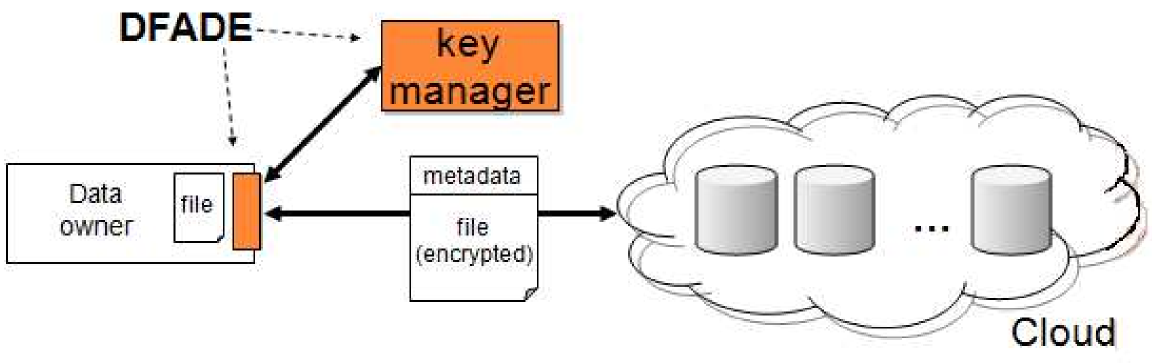


Figure 4.1 The DFADE system. Each client interacts with one or multiple key managers and uploads / downloads data files to / from the cloud.

Figure 4.1 illustrates an overview of the DFADE system. The cloud hosts data files on behalf of a group of DFADE users who want to outsource data files to the cloud based on their definitions of file access policies. DFADE can be viewed as an overlay system atop the underlying cloud. It applies security protection to the outsourced data files before they are hosted on the cloud.

5. DFADE SYSTEM ARCHITECTURE



E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

Figure 5.1 The DFADE System Architecture

5.1 Entities

As shown in Fig. 5.1, the DFADE system is composed of two main entities:

- **DFADE clients:** A DFADE client (or client for short) is an interface that bridges the data source (e.g. file system) and the cloud. It applies encryption (decryption) to the outsourced data files uploaded to (downloaded from) the cloud. It also interacts with the key managers to perform the necessary cryptographic key operations.

- **Key managers**: DFADE is built on a quorum of key managers [19], each of which is a stand-alone entity that maintains policy-based keys for access control and assured deletion.

The cloud, maintained by a third-party provider, provides storage space for hosting data files on behalf of different DFADE clients in a pay-as-you-go manner. Each of the data files is associated with a combination of file access policies. DFADE is built on the thin-cloud interface [15], and assumes only the basic cloud operations for uploading and downloading data files. We emphasize that we do not require any protocol and implementation changes on the cloud to support DFADE.

5.2 Deployment

In our current design, a DFADE client is deployed locally with its corresponding data source as a local driver or daemon. Note that it is also possible to deploy the DFADE client as a cloud storage proxy [1], so that it can interconnect multiple data sources. In proxy deployment, we can use standard TLS/SSL [9] to protect the communication between each data source and the proxy.

In DFADE, the set of key managers is deployed as a centralized trusted service, whose trustworthiness 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 DFADE for its employees. We note that this centralized control is opposed to the core design of Vanish [12], which proposes to use decentralized key management on top of existing P2P DHT systems. However, as discussed in Section 2, there is no straightforward solution to develop fine-grained cryptographic key management operations over a decentralized P2P DHT system. Also, the Vanish implementation that was published in [12] is subject to Sybil attacks [18], which particularly target DHT systems. In view of this, we propose to deploy a centralized key management service, and use a quorum scheme to improve its robustness.

5. 3 Cryptographic Keys

DFADE defines three types of cryptographic keys to protect data files stored on the cloud:

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

- **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.

- **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 DFADE client that is authorized to access files of the associated policy. The access key is built on attribute-based encryption [7], 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 DFADE. The following explains how we manage such keys to achieve our security goals.

5. 4 Security Goals

We formally state the security goals that DFADE seeks to achieve in order to protect the outsourced data files.

- **Threat model.** Here, we consider an adversary that seeks to compromise the privacy of two types of files that are outsourced and stored on the cloud: (i) *active files*, i.e., the data files that the adversary is unauthorized to access and (ii) *deleted files*, i.e., the data files that have been requested to be deleted by the authorized parties. Clearly, FADE needs to properly encrypt outsourced data files to ensure that their information is not disclosed to unauthorized parties. The underlying assumption is that the encryption mechanism is secure, such that it is computationally infeasible to recover the encrypted content without knowing the cryptographic key for decryption.

5.4.1 Security properties: Given our threat model, we focus on two specific security goals that DFADE seeks to achieve for fine-grained security control:

- **Policy-based access control.** A FADE client is authorized to access only the files whose associated policies are active and are satisfied by the client.
- **Policy-based assured deletion.** A file is deleted (or permanently inaccessible) if its associated policies are revoked and become obsolete. That is, even if a file copy that is associated with revoked policies exists, it remains encrypted and we cannot retrieve the corresponding cryptographic keys to recover the file. Thus, the file copy becomes unrecoverable by anyone (including the owner of the file).

6. MATHEMATICAL MODULES

We now introduce the basic operations of how a client Uploads/downloads files to/from the cloud. We start with the case where each file is associated with a single policy, and then explain how a file is associated with multiple policies.

6.1 File Upload:

```
System=<Input, Output, Process>
Input:
{
    No. of Files to be processed
}
Output:
{
    No. of files uploaded successfully
}
 Process :{ Key generation}
```

Figure 6.1 shows the file upload operation. The client first requests the public control key (ni, ei) of policy Pi from the key manager, and caches (ni, ei) for subsequent uses if the same policy Pi is associated with other files. Then the client generates two random keys K and Si, and sends {K} Si, Seii, and {F} K to the cloud2. Then the client must discard K and Si. To protect the integrity of a file, the client computes an HMAC signature on every encrypted file and stores the HMAC signature together with the encrypted file in the cloud. We assume that the client has a long-term private secret value for the HMAC computation. Si, and decrypt {K} Si and hence {F} K.

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

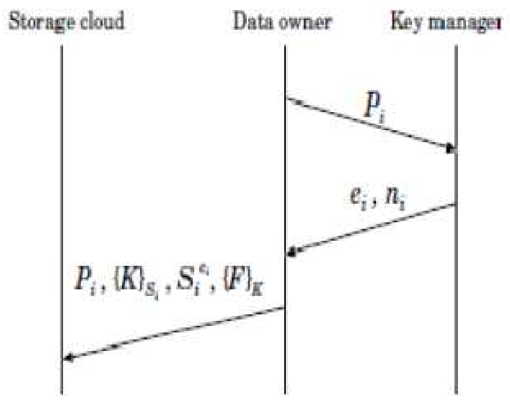


Figure 6.1 File Upload Operation

Turing machine:

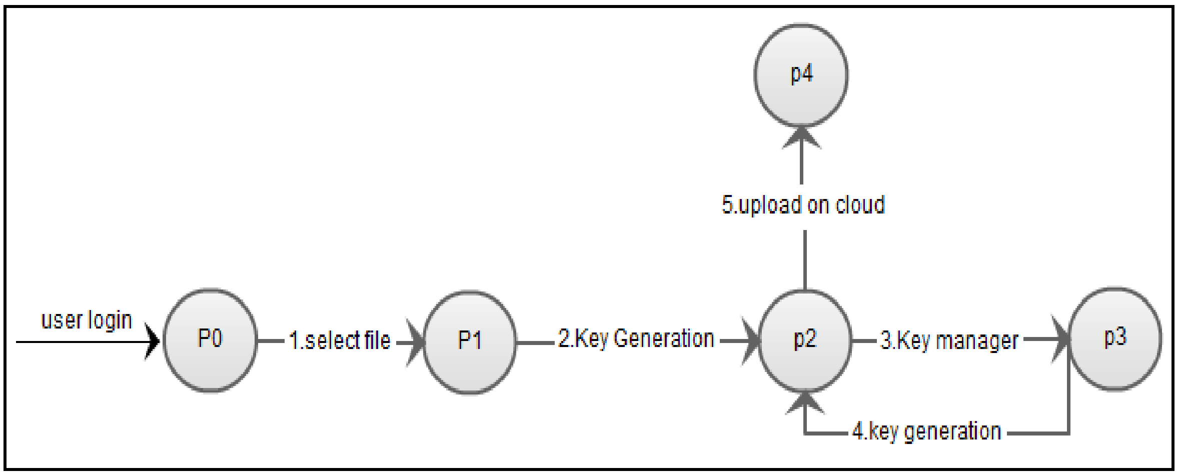


Figure 6.2 Turing Machine for File Upload Operation

6.2 File Download:

System=<Input, Output, Process>
Input:
{
No. of Files to be processed
}
Output:
{
No. of files downloaded successfully
}
Process :{ Key generation}

Figure 6.3 shows the file download operation. The client fetches {K} Si, Seii, and {F} K from the cloud. The client will first check whether the HMAC signature is valid before decrypting the file. Then the client generates a secret random number R, computes Rei, and sends Seii ·Rei = (SiR) ei to the key manager to request for decryption. The key manager then computes and returns ((SiR) ei) di = SiR to the client, which can now remove R and obtain Si, and decrypt {K} Si and hence {F} K.

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

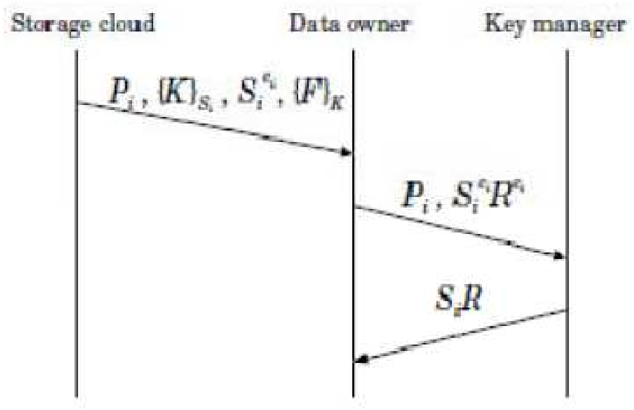


Figure 6.3 File Download Operation

Turing machine:

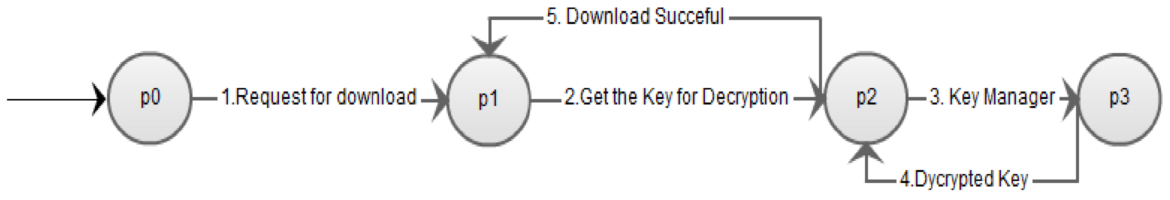


Figure 6.4 Turing Machine for File Download Operation

7. DESIGN PROCESS

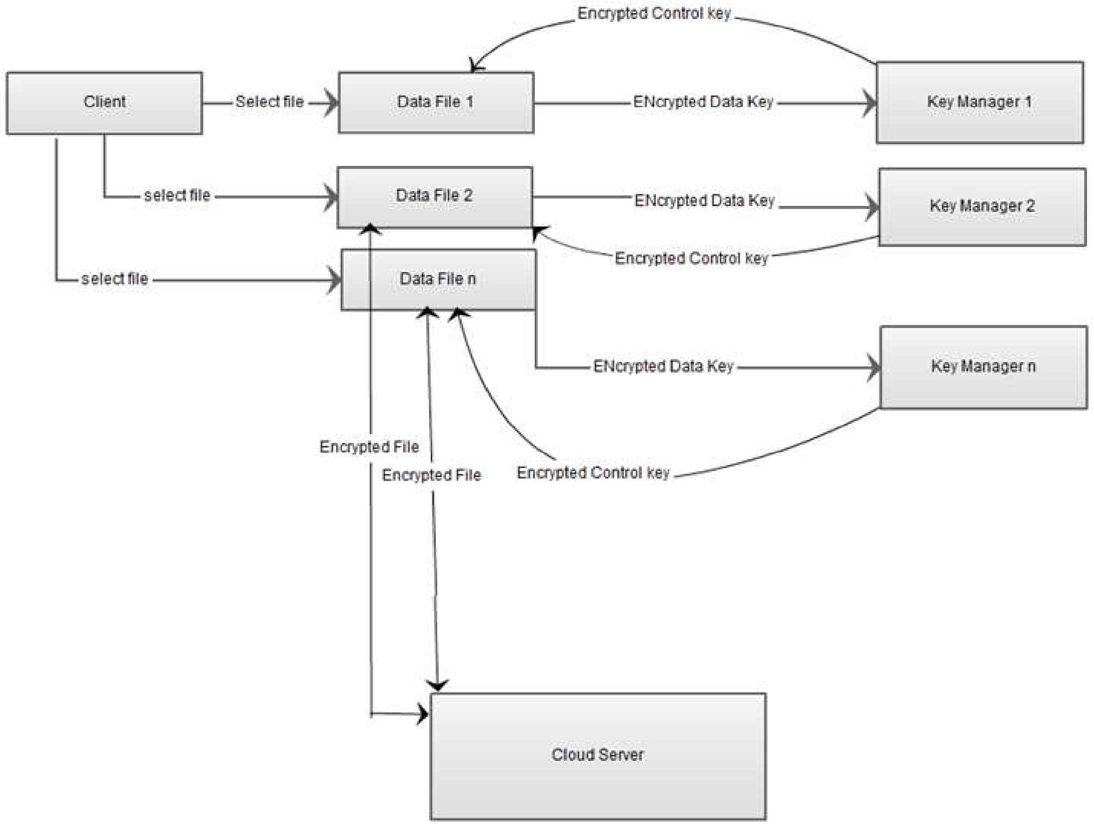


Figure 7.1: Process flow of access control and assured deletion

7.1 Policy-based access control

A FADE client is authorized to access only the files whose associated policies are active and are satisfied by the client. It gives secrete key to the end user for file uploading and downloading.

7.2 Policies Renewal

Is the term related to the access permission's wherein a user requests to the cloud manager to provide the policies other than which are being allotted to him/her. For the blocked user's(Fraud) in order to have access to the resources stored in the cloud server need's to have access permission's which are being provided by the cloud manager when the blocked user goes for requesting the files.

7.3 Policy-based assured deletion:

A file is deleted (or permanently inaccessible) if its associated policies are revoked and become obsolete. That is, even if a file copy that is associated with revoked policies, it remains encrypted and we cannot retrieve the corresponding cryptographic keys to recover the file. Thus, the file copy becomes unrecoverable by anyone (including the owner of the file).

8. OBSERVATIONS

We first measure the time performance of our DFADE Prototype. In order to identify the time overhead of DFADE, we divide the running time of each measurement in to three components:

- File transmission time, the uploading/downloading time for the data file between the client and the Cloud.
- Metadata transmission time, the time for uploading/Downloading the metadata, which contains the Policy information and the cryptographic keys associated. With the file, between the client and the Cloud.

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

- Cryptographic operation time, the total time for cryptographic operations, this includes the total computational time used for performing AES and HMA Con the file, and the time for the client to coordinate with the quorum of key managers on operating the cryptographic keys.

9. EVALUATION

We now evaluate the empirical performance of our implemented prototype of DFADE atop Amazon S3. It is crucial that DFADE does not introduce substantial performance or monetary overhead that will lead to a big increase in data management costs. In addition, the cryptographic operations of DFADE should only bring insignificant computational overhead. Therefore, our experiments aim to answer the following questions: What is the performance and monetary overhead of DFADE? Is it feasible to use DFADE to provide file assured deletion for cloud storage? Our experiments use Amazon S3 APAC servers that reside in Singapore for our cloud storage backend. Also, we deploy the client and the key managers within a departmental network. We evaluate DFADE on a per-file basis, that is, when it operates on an individual file of different sizes. We can proportionally scale our results for the case of multiple files.

9.1 TIME PERFORMANCE OF FADE

We first measure the time performance of our DFADE prototype. In order to identify the time overhead of DFADE, we divide the running time of each measurement into three components:

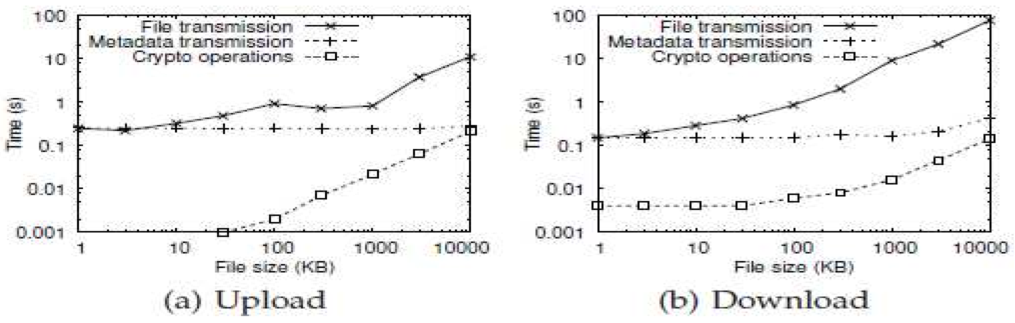
1. File transmission time, the uploading/downloading time for the data file between the client and the cloud.
2. Metadata transmission time, the time for uploading/ downloading the metadata, which contains the policy information and the cryptographic keys associated with the file, between the client and the cloud.
3. Cryptographic operation time, the total time for cryptographic operations, which includes the total computational time used for performing AES and HMAC on the file, and the time for the client to coordinate with the quorum of key managers on operating the cryptographic keys. We average each of our measurement results over 10 different trials.

9.1.1 Evaluation of Basic Design

We evaluate the time performance of the basic design of DFADE, in which we use a single key manager and do not involve ABE.

Performance of file upload/download operations:

In this experiment, we measure the running time of the file upload and download operations for different file sizes (including 1KB, 3KB, 10KB, 30KB, 100KB, 300KB, 1MB, 3MB, and 10MB).



9.1 Performance of file upload / download operations

Figure 9.1 shows the results. First, the cryptographic operation time increases with the file size, mainly due to the symmetric-key encryption applied to a larger file. Nevertheless, we find that in all cases of file upload / download operations, the time of cryptographic operations is no more than 0.2s (for a file size within 10MB), and accounts for no more than 2.6% of the file transmission time. We expect that DFADE only introduces a small time overhead in cryptographic operations as compared to the file transmission time, where the latter is inevitable even without DFADE.

Also, the metadata transmission time is always around 0.2s, regardless of the file size. This is expected, since the metadata file only stores the policy information and cryptographic keys, both of which are independent of the data files. The file transmission time is comparable to the metadata transmission time for small files. However, for files larger than 100KB, the file transmission time becomes the dominant factor. For instance, to upload or download a 10MB file the sum of the metadata transmission time and the cryptographic operation time (both are due to DFADE) account for 4.1% and 0.7% of the total time, respectively.

Note that the upload and download operations are asymmetric and use different times to complete the operations. Nevertheless, the performance overhead of DFADE drops when the size of the data file being protected is large enough, for example, on the megabyte scale.

10. CONCLUSION

We propose a cloud storage system called DFADE, 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 DFADE to demonstrate its practicality, and empirically study its performance overhead when it works with Amazon S3. Our experimental results provide insights into the performance-security trade-off when DFADE is deployed in practice.

REFERENCES

[1] Yang Tang, Patrick P. C. Lee, John C. S. Lui, Radia Perlman, “Secure Overlay Cloud Storage with Access Control and Assured Deletion”, IEEE TRANSACTIONS AND DEPENDABLE AND SECURE COMPUTING VOL.9 NO.6 ,2012.
[2] Amazon. Case Studies. http://aws.amazon.com/solutions/casestudies/#backup.
[3] Amazon. Smug Mug Case Study: Amazon Web Services. http://aws.amazon.com/solutions/case-studies/smugmug/, 2006.
[4] Amazon S3. http://aws.amazon.com/s3, 2010.
[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A View of Cloud Computing. *Comm. of the ACM*, 53(4):50–58, Apr 2010.
[6] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik. Scalable and Efficient Provable Data Possession. In *Proc. of SecureComm*, 2008.
[7] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *Proc. of IEEE Symp. on Security and Privacy*, May 2006.
[8] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based Encryption with Efficient Revocation. In *Proc. of ACM CCS*, 2008.
[9] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2, Aug 2008. RFC 5246.
[10] Dropbox. http://www.dropbox.com, 2010.
[11] R. Geambasu, J. P. John, S. D. Gribble, T. Kohno, and H. M. Levy. Keypad: Auditing File System for Mobile Devices. In *Proc. Of EuroSys*, April 2011.
[12] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. In *Proc. Of USENIX Security Symp.*, Aug 2009.
[13] 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.
[14] P. Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proc. of USENIX Security Symp.*, 1996.
[15] JungleDisk. http://www.jungledisk.com/, 2010.
[16] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable Secure File Sharing on Untrusted Storage. In *Proc. of USENIX FAST*, 2003.
[17] S. Kamara and K. Lauter. Cryptographic Cloud Storage. In *Proc. of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization*, 2010.
[18] LibAWS++. http://aws.28msec.com/, 2010.
[19] A. J. Menezes, P. C. van Oorschot, and S. A.Vanstone. *Handbook of Applied Cryptography*. CRC Press, Oct 1996.
[20] S. Nair, M. T. Dashti, B. Crispo, and A. S. Tanenbaum. A Hybrid PKI-IBC Based Ephemerizer System. *IFIP International Federation for Information Processing*, 232:241–252, 2007.
[21] Nasuni. Nasuni Announces New Snapshot Retention Functionality in Nasuni Filer; Enables Fail-Safe File Deletion in the Cloud, Mar 2011. http://www.nasuni.com/news/press-releases/nasuniannounces-new-snapshot-retention-functionality-in-nasuni-filerenables-fail-safe-file-deletion-in-the-cloud/.

E-ISSN: 2321–9637
Volume 1, Issue 5, December 2013
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

[22]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.
[23] R. Perlman. File System Design with Assured Delete. In *ISOC NDSS*, 2007.
[24] R. Perlman, C. Kaufman, and R. Perlner. Privacy-Preserving DRM. In *IDtrust*, 2010.
[25] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Attribute-Based Systems. In *Proc. of ACM CCS*, 2006.
[30] W. Wang, Z. Li, R. Owens, and B. Bhargava. Secure and Efficient Access to Outsourced Data. In *ACM CCSW*, Nov 2009.
[31] W. Stallings. *Cryptography and Network Security*. Prentice Hall, 2006.
[32] 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.
[33] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute Based Data Sharing with Attribute Revocation. In *Proc. of ACM ASIACCS*, Apr 2010.
[34] M. Vrable, S. Savage, and G. M. Voelker. Cumulus: Filesystem backup to the cloud. *ACM Trans. on Storage*, 5(4), Dec 2009.