Enabling Data Dynamics and Achieving Access Control and Assured Deletion in Cloud

Nilam Musale[1], Vilas Gaikwad[2]

[1]Computer Engineering Department, JSPMNTC RSSOER, Savitribai Phule Pune University,

[2]Assistant Professor, Computer Engineering Department, JSPMNTC RSSOER, Savitribai Phule Pune University

Abstract: ***In this paper, a new practical cloud storage framework is proposed which is build on FADE system. FADE is cloud storage system which is a practical, implementable and readily deployable. It protects deleted data with policy-based file assured deletion. FADE system is using cryptographic techniques in which it first encrypts the outsourced data files to maintain their privacy and integrity. It also assuredly deletes files to make them unrecoverable to anyone upon revocation of file access policies. A working prototype of FADE is implemented on cloud storage services. We propose two extensions to the basic design of FADE. The first extension is to use attribute based encryption (ABE) to authenticate clients through policy based access control. The second extension is to add third dynamic operation to basic design of FADE architecture known as UPDATE operation.***

Keywords: access control, assured deletion, backup/recovery, cloud storage.

1. Introduction

Cloud storage is a model of networked enterprise storage where data is stored in virtualized pools of storage which are generally hosted by third parties. Hosting companies operate large data centers, and people who require their data to be hosted buy or lease storage capacity from them. The data center operators, in the background, virtualizes the resources according to the requirements of the customer and expose them as storage pools, which the customers can themselves use to store files or data objects. Physically, the resource may span across multiple servers and multiple locations. The safety of the files depends upon the hosting companies, and on the applications that leverage the cloud storage.

Cloud storage services may be accessed through a web service application programming interface (API) or by applications that utilize the API, such as cloud desktop storage, a cloud storage gateway or Web-based content management systems.

Security issues

Security issues become relevant as we now outsource the storage of possibly confidential data to third parties. In this paper, we are particularly interested in two security issues.

1) It need to provide guarantees of access control, in which we must ensure that only authorized parties, can access the outsourced data on the cloud. In particular, we must prohibit third-party cloud storage providers from mining any sensitive information of their clients' data for their own marketing purposes.
2) 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. Keeping data permanently is undesirable, as data may be unexpectedly disclosed in the future due to malicious attacks on the cloud or careless management of cloud operators. The challenge of achieving assured deletion is that we have to trust cloud storage providers to actually delete data, but they may be reluctant in doing so. Also, cloud storage providers typically keep multiple backup copies of data for fault-tolerance reasons. It is uncertain, from cloud clients' perspectives, whether cloud providers reliably remove all backup copies upon requests of deletion.

The security issues motivate us, as cloud clients, to have a system that can enforce access control and assured deletion of outsourced data on the cloud in a fine-grained manner. However, building such a system is a difficult thing, especially when it involves protocol or hardware modifications in cloud storage infrastructures that are externally owned and managed by third-party cloud providers. Thus, it is required to design a secure overlay cloud storage system that can be overlaid and work smoothly atop existing cloud storage services.

2. Literature Survey

In [1], authors have presented policy based assured deletion concept. Every file to be stored on cloud is associated with some policy. Any user (including owner)is not able to access the file after policy associated with file expires. The FADE system performs cryptographic operations on file.

In [2], author has proposed a new methodology for recognizing Ciphertext-Policy Attribute Encryption (CP-ABE) under concrete and non-interactive cryptographic assumptions in the standard model. The proposed solutions allow any encryptor to specify access control in terms of any access formula over the attributes in the system.

In [3], author has proposed Vanish, a system to ensure that all copies of certain data become unreadable after a user-specified time, without any specific action on the part of a user, and even if an attacker obtains both a cached copy of that data and the user's cryptographic keys and passwords. The proposed system achieves this challenge through a novel integration of cryptographic techniques with global-scale, P2P, distributed hash tables (DHTs).

In [4], author has constructed a highly efficient and provably secure PDP (Provable Data Possession) technique based on symmetric key cryptography, while not requiring any bulk encryption. Also, in contrast with its predecessors, our PDP technique allows outsourcing of dynamic data, i.e, it efficiently supports operations, such as block modification, deletion and append.

In [5], author has developed a new cryptosystem for fine-grained sharing of encrypted data known as Key-Policy Attribute-Based Encryption (KP-ABE). In proposed cryptosystem, cipher-texts are labeled with sets of attributes and private keys are associated with access structures that control which cipher-texts a user is able to decrypt.

3. Existing System

The existing system FADE[1] support basic cryptographic operations as follows:

The client during FADE implementation uses four function calls to enable end users to interact with the cloud:

• **Upload (file, policy)**. The client encrypts the input file according to the associated policy (or a Boolean combination of policies).The file is encrypted using the 128-bit AES algorithm with the cipher block chaining (CBC) mode. After encryption, 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.

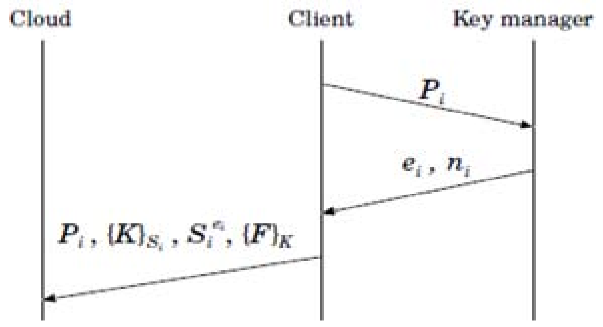

Figure 1: File Upload

• **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

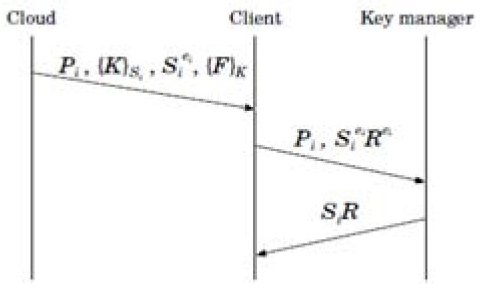

Figure 2: File Download

• **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.

• **Renew (file, new_policy).** The client first 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.

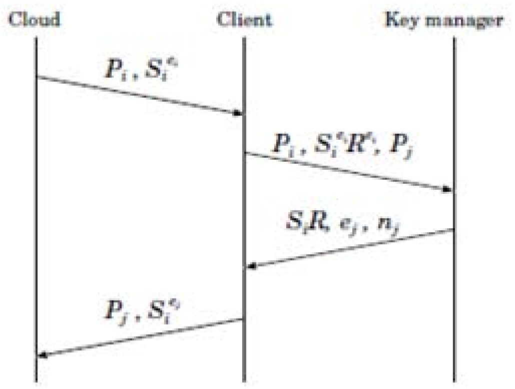

Figure 3: A special case of policy renewal - when policy Pi is renewed to policy Pj .

Drawback of Existing System

1) Existing system does not support update operation.
2) Existing system is not scalable if we increase number of supported clients and their association with policies.

4. Proposed System

The proposed system FADE allows user to update the file as well as it supports *Access Control with ABE*.

Proposed System Architecture

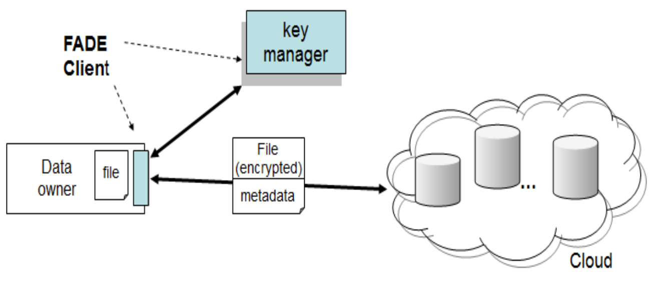

Figure 4: FADE Architecture

Ciphertext Policy Attribute Based Encryption (CP-ABE)

When a client needs to recover a file from the cloud, a client needs to request the key manager (assuming that only a single key manager is deployed) to decrypt the data key [2]. The client needs to provide authentication credentials to the key manager to show that it fulfills the policies associated with the files. One method for this authentication process is based on the public-key infrastructure (PKI).But, this client-based authentication requires the key manager to have accesses to the association of *every* client and its satisfied policies. This puts limits on the scalability and flexibility if we scale up the number of supported clients and their associations with policies. To address the scalability issue, *attribute -based encryption (ABE)* is found out to be the right solution.

An (Key-Policy) Attribute Based Encryption scheme consists of four algorithms.

Setup

This is a randomized algorithm that takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

Encryption

This is a randomized algorithm that takes as input a message m, a set of attributes S, and the public parameters PK. It outputs the ciphertext E.

Key Generation

This is a randomized algorithm that takes as input an access structure A, the master key MK and the public parameters PK. It outputs a decryption key D.

Decryption

This algorithm takes as input the ciphertext E that was encrypted under the set of attributes, the decryption key D for access control structure A and the public parameters PK. It outputs the message M .

5. Mathematical model of Proposed system

The system is modeled as S = {s, e, A, P, X, Y, F, Fc |ϕ}
Where, s is the initial state
The user will input the File (*F*)
Attributes (A) and Policy(P)
e is the end state of the system which comprise of two states :
If Policy is Active: The file will be accessible.
If Policy Expires: The file will not be accessible.
X = Set of inputs in the system
X = {F,*P*,*k* }
k = encryption key
P= Policy associated with file.
Y = Set of outputs
Y = {F'}
F' = Retrieved file after decryption from various nodes using key (*k*)
Function Fc = {Enck(D), Decrk(E)}
Where, Enck(D) = Encryption of data using key (k) for storing data in encrypted format
Deck(E) = Decryption of data for retrieving original data
Φ = Constraint on whole system.

6. Methodology

New proposed FADE seeks to achieve both access control and assured deletion for outsourced data. The design of FADE is centered around the concept of policy-based file assured 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. 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.

7. Results

We have implemented our system on Jelastic cloud service provider. The system is developed using Java framework (version jdk 1.7) on Windows os.The Netbean (version 7.1) is used as a development tool. For back end we are using MySQL. The system does not require any specifc hardware to run, any standard platform is capable of running the application.

We have implemented our system on local desktop system as well as on Jelastic cloud service provider. We plotted time versus file uploading operation in existing system graph.While in proposed system we have plotted time versus file update operation.

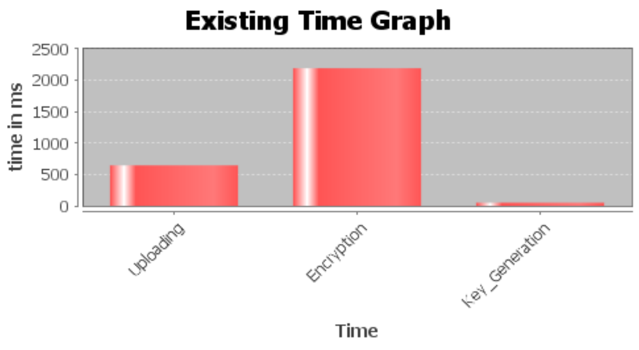

Figure 5: Performance of Existing System

We come to know that if a user wants to update existing system on cloud then he can do it in less time as there no need of key generation again.Therefore,with update operation proposed by our system time required is less.If update operation is not there and suppose user wants to upload file then he has to delete file first on cloud and again upload the new file with key generation operation and other cryptographic operation.Therefore,with our proposed dynamic UPDATE operation in existing FADE system we can save time.

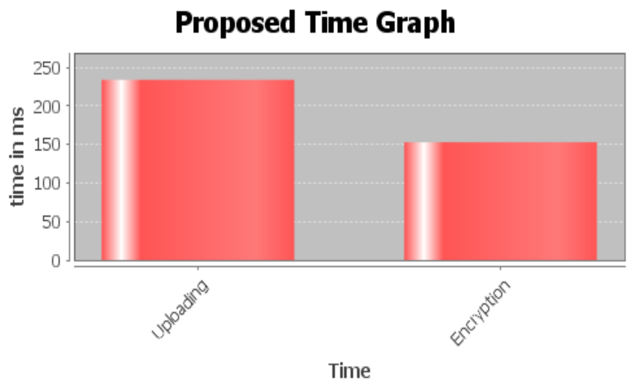

Figure 6: Performance of Proposed System

8. Conclusion

We have proposed extension to the basic design of FADE system. In FADE system, access control and assured deletion for files is achieved. The file which are to be stored on cloud are first encrypted and then stored on cloud. A prototype of FADE is implemented on cloud service provider to demonstrate its practicality. The proposed system FADE support dynamic operations such as file upload, download and update. Experimental result exhibit various performance trade-off when FADE is deployed in practice.

9. Future Enhancement

In future work, user should be able to append data to existing file stored on cloud. In our current implementation, the operations of FADE are on a per-file basis, such that each data file has one corresponding policy metadata file. To reduce the metadata overhead of FADE, we can associate a batch of multiple data files (e.g., files under the same directory) with the same policy metadata and the same set of cryptographic keys (including the data key and the control keys of policies). The advantage of the batch-based approach is that we can use one single policy metadata for multiple data files. Thus, if the data files are of small size, then the batch-based approach can reduce the storage overhead due to the policy metadata.

References

[1] Yang Tang, Patrick P. C. Lee, "Secure Overlay Cloud Storage with Access Control And Assured Deletion," IEEE transactions and dependable and secure computing. Vol.9 No. 6 2012.
[2] J. Bethencourt, A. Sahai and B. Waters, "Cipher text-Policy Attribute Based Encryption," In proc. Of Symp. On Security and Privacy, May 2006
[3] R. Geambasu,T. Kohno, A. Levy, H. M. Levy."Vanish: Increasing Data Privacy with Self-Destructing Data", In Proc. Of USENIX Security Symp.., Aug 2009.
[4] G. Ateniese, R. D. Pietro, L. V. Mancini, "Scalable And Efficient Provable Data Possesion," In. Proc. Of SecureComm, 2008
[5] 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.

Author Profile

Nilam Musale received the B.E. degrees in Computer Science And Engineering from Dattajirao Kadam Textile Institute Of Engineering And Technology, Ichalkaranji in 2011. She is now pursuing ME in Computer Engineering from JSPM Narhe Technical Campus,Pune.