International Journal For Technological Research In Engineering
Volume 1, Issue 4, December-2013 *ISSN (Online): 2347 - 4718*

SECURE OVERLIE CLOUD STORAGE WITH ACCESS CONTROL AND GUARANTEED REMOVAL USING FADE

R. Manikandasamy[1], K. K. Kanagamathanmohan[2], Dr. C. Kumar Charlie Paul[3]
[1, 2] A.S.L Pauls College of Engineering & Technology, Coimbatore, India.
[3] Department of Computer science, Anna University Chennai, India.

***Abstract*: Now we are using 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. 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. 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 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 off of performance overhead. Work provides insights of how to incorporate value added security features into current data outsourcing applications**

***Keywords:* Fade, Decentralized Erasure Code, Proxy Re-Encryption, Threshold Cryptography, Secure Storage System.**

I. INTRODUCTION

Cloud computing or something being in the cloud is an appearance used to describe a variety of different types of computing concepts that involve a large number of computers connected through a real time communication network such as the Internet. In knowledge cloud computing is a synonym for distributed computing over a network and means the ability to run a program on many connected computers at the same time.

The expression is also more normally used to refer to network-based services which appear to be provided by real server hardware which in information are served up by effective hardware simulated by software running on one or more real machines. Such virtual servers do not physically exist and can therefore be moved around and scaled up on the fly without affecting the end user questionably rather like a cloud. In this Fig 1.1 cloud also focuses on maximizing the effectiveness of the shared resources.

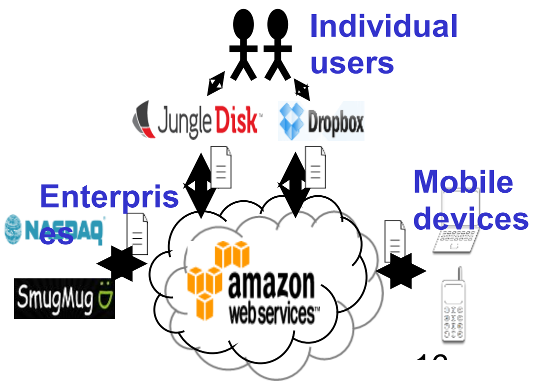

Fig.1. A cloud Network

Cloud resources are usually not only shared by multiple users but are also dynamically. Reallocated per demand. This can work for allocating resources to users. For example SmugMug a photo distribution website chose to host terabytes of photos on Amazon S3 in 2006 and saved thousands of dollars on maintaining storage devices.

In particular, with the advent of smart phones be expecting that more people will use Dropbox like tools to move audio/video files from their smart phones to the cloud specified that smart phones typically have limited storage resources. Nevertheless security concerns become relevant as we now outsource the storage of possibly sensitive data to third parties.

Particularly interested in two security issues Initial need to provide guarantees of access manage in which we must ensure that only approved parties can access the outsourced data on the cloud. In Fig 1.2 particular must prohibit third party cloud storage providers from mining any sensitive information of their client's data for their own marketing purposes. Subsequently it is important to provide guarantees of confident deletion meaning that outsourced data is permanently unapproachable to anybody (including the data owner) upon requests of deletion of data.

Keeping data permanently is unwanted 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 unwilling in doing so. Also cloud storage providers typically keep multiple backup copies of data for fault tolerance reasons. It is unsure from cloud client's perspectives whether cloud providers dependably remove all backup copies upon requests of deletion.

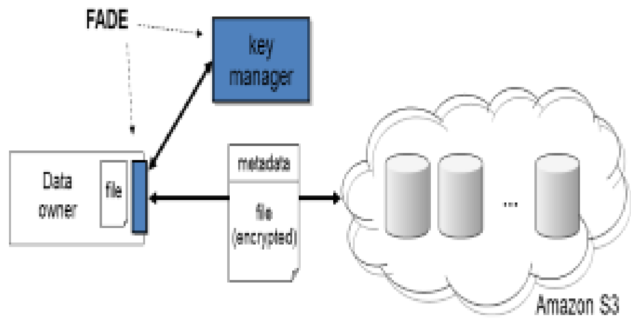

Fig.2. FADE Access Diagram

Present FADE a secure overlie cloud storage system that provides fine-grained access control and assured deletion for outsourced data on the cloud while operational seamlessly atop today's cloud storage services.

In FADE lively data files that remain on the cloud are associated with a set of user defined file access policies (e.g., time expiration read/write permissions of approved users) such that data files are accessible only to users who satisfy the file access policies. In addition FADE generalize time based file confident deletion (i.e., data files are assuredly deleted upon time expiration)

II. OVERVIEW OF RESEARCH WORK

Provide value added security features into today's cloud storage services. Introduce policy based file guaranteed 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 treatment operations with the emphasis on fine grained security protection.

On top of policy based file assured deletion and design and implement two new features 1) fine grained access control based on attribute based encryption and 2) fault tolerant key management with a quorum of key managers based on threshold secret sharing. Subsequently it is important to provide guarantees of confident deletion meaning that outsourced data is permanently unapproachable to anybody (including the data owner) upon requests of deletion of data. Policy-based file assured deletion scheme that reliably deletes files with regard to revoked file access policies. In this context design the key management schemes for various file manipulation operations with the emphasis on fine-grained security protection.

On top of policy-based file assured deletion design and implement two new features 1) fine-grained access control based on attribute-based encryption and 2) fault-tolerant key management with a quorum of key managers based on threshold secret sharing.
Implement a working prototype of FADE atop Amazon S3. Implementation of FADE exports a set of APIs that can be adapted into different data outsourcing applications.

Implement a working prototype of FADE atop Amazon S3.completion of FADE exports a set of APIs that can be adapted into different data outsourcing applications. Empirically evaluate the performance overhead of FADE atop Amazon S3. Using experiments in a realistic network situation show the feasibility of FADE in improving the security protection of data storage on the cloud in practice. Also analyze the monetary cost overhead of FADE under a practical cloud backup scenario.

III. ORGANIZATION OF THE REPORT

The rest of the thesis is organized as follows. Chapter two describes the related work addressing code decompression for embedded system. Chapter three analyzes about the existing Policy based file assured Deletion Cryptographic key techniques. Chapter 4 describes software specification. Chapter five discuss about software organization. Chapter six illustrates the implementation and techniques and result in this project. Chapter seven conclude and future work.

IV. POLICY-BASED FILE ASSURED DELETION

A. Introduction

FADE seeks to achieve both access control and assured deletion for outsourced data. The design of FADE is centered approximately the concept of policy-based file assured deletion. Initial review time based file assured deletion proposed in earlier work. Then explain the more general concept policy based file assured deletion and motivate why it is important in certain scenarios.

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 removal the file content is encrypted with a information key and the data key is further encrypted with the control key corresponding to the policy. When the rule is revoked the matching control key will be removed from the key manager. Thus when the strategy connected with a file is revoked and no longer holds the data key and hence the encrypted content of the file cannot be well again with the control key of the policy.

B. Participants In The System

Our system is composed of three participants the information owner the key manager and the storage cloud. They are described as follows. 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 consumer stage program, a mobile machine or even in the form of a plug-in of a client application.

The key manager maintains the policy-based control keys that are used to encrypt data keys. It respond to the statistics owner's requests by performing encryption decryption regeneration and revocation to the control keys.

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

C. Cryptographic Keys

FADE defines three types of cryptographic keys to protect data files stored on the cloud. A data key is a random secret that is generated and maintained by a FADE client. It is second-hand for encrypting or decrypting data files via symmetric key encryption (e.g., AES).

A control key is associated with a meticulous policy. It is represented by a community private key pair and the private control key is maintained by the quorum of key managers. It is second-hand to encrypt/decrypt the data keys of the files protected with the same policy. The control key forms the foundation of rule based assured deletion.

Similar to the control key an access key is associated with a particular policy and is represented by a public-private key pair. 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 Fig 3.1 is built on attribute based encryption and forms the basis of policy based access control.

- File protected with data key
- Data key protected with control key

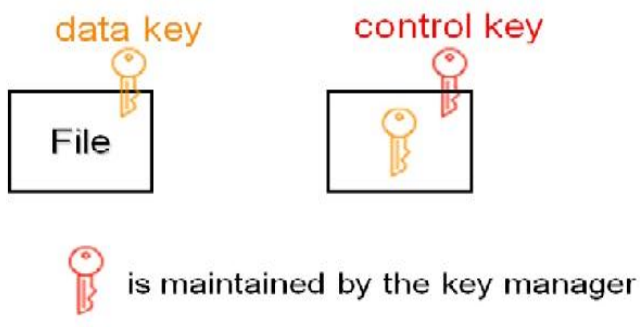

Fig.3. Key operation

When a policy is revoked the control key is uninvolved. The encrypted figures key and consequently the encrypted file cannot be well again

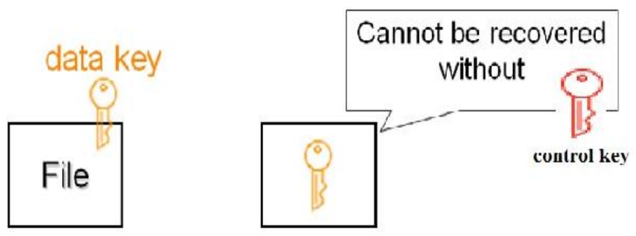

Fig.4. Control Key operation.

The file is deleted i.e., Fig 3.2 even a copy exist it is encrypted and inaccessible by everyone.

Successfully decrypt an encrypted file stored on the cloud requires the accurate data key control key and access key. Devoid of any of these keys it is computationally infeasible to recover an outsourced file being protected by FADE. The subsequent explains how we administer such keys to achieve our security goals.

D. 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 cannot recover Si from Si and hence cannot recover K and the file F. That the file F which is tied to policy Pi is confidently deleted. Note that the strategy revocation operations do not involve interactions with the storage cloud.

E. Multiple Policies

In addition to one policy per file FADE supports a Boolean combination of multiple policies. 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 fulfilled then the data is reachable.

Suppose that F is associated with conjunctive policies P1 ∧ P2 ∧ · · · ∧ Pm. To upload F to the storage cloud, the data owner first randomly generates a data key K, and secret keOn 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. Suppose that F is associated with disjunctive policies Pi1 ∨ Pi2 ∨ · · · ∨ Pim. To upload F to the cloud the data owner will send the Tang et al.

Following: {K}S1 , {K}S2 , . . ., {K}Sm, Se11 , Se22 , . . ., Semm , and {F}K. Therefore, the data owner needs to compute m different encrypted copies of K. On the other hand over to recover F can use any one of the policies to decrypt the file, as in the above operations. To delete a file associated with conjunctive policies simply revoke any of the policies (say, Pj). Cannot recover Sj and hence the data key K and file F. On the other hand, to delete a file associated with disjunctive policies, we need to revoke all policies, so that S ejj cannot be recovered for all j. letter that for any Boolean grouping of policies Express it in canonical form e.g. in the disjunction (OR) of conjunctive (AND) policies.

F. The Fade Architecture

Implement a working prototype of FADE JAVA and we use the Open SSL library for the cryptographic operations. In Fig 3.3 addition 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. Figure shows the FADE architecture. In the following define the metadata of FADE attached to individual files. Then describe how we implement the data owner and the key manager and how the data owner interacts with the storage cloud.

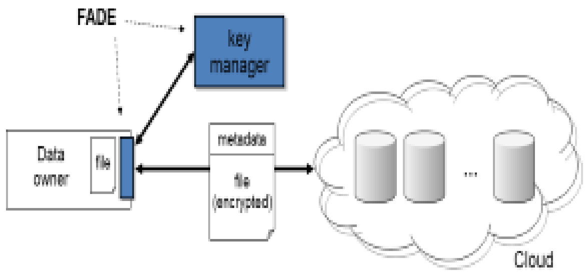

Fig.5. Architecture of FADE

Policy metadata- The policy metadata includes the specification of the Boolean combination of policies and the corresponding encrypted cryptographic keys. Assume that each single policy is specified by a unique 4-byte integer identifier. To symbolize a Boolean grouping 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 (which is the expected behavior of FADE). In other language for two data files that are under the same rule 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. In Section 5 we discuss how we may associate the same policy metadata file with multiple data files so as to reduce the metadata overhead.

G. Data Owner And Storage Cloud

Implementation of the data owner uses the following four function calls to enable end users to interact with the storage cloud.

Upload (file, policy): The data owner encrypts the input file using the specified policy (or a Boolean combination of policies). Our goal is to show the practicality of FADE when it is deployed with today's cloud storage services. This enables us to renew policies directly on the policy metadata without retrieving the entire file from the storage cloud. It then sends the encrypted file and the metadata onto the cloud. In our implementation the file is encrypted using the 128-bit AES algorithm with the cipher block chaining (CBC) mode yet we can adopt a different symmetric key encryption algorithm depending on applications.

The data owner retrieves the file and the policy metadata from the cloud checks the integrity of the file and decrypts the file. The data owner tells the key manager to permanently revoke the specified policy. All files connected with the strategy 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. Lastly it sends the policy metadata back to the cloud.

V. MODE OF IMPLEMENTATION

Implement a working prototype of FADE using JAVA. Our implementation is built on off-the-shelf library APIs. Specifically use the Open SSL library for the cryptographic operations the cpabe library for the ABE-based access control and the ssss library for sharing control keys to a quorum of key managers. The ssss documentation is initially calculated as a command line utility to deal with keys in ASCII format. Slightly modify ssss and add two functions to split and combine keys in binary arrangement, so as to make it well-matched with other libraries. In addition use Amazon S3 as our cloud storage backend. In the following define the metadata of FADE being attached to individual data files. We then describe how implement the client and a quorum of key managers and how the client interacts with the cloud.

A. Blinded RSA Algorithm

RSA involves a control key. The control key is used for encrypting Data key. Data key encrypted with the control key can only be decrypted in a reasonable amount of time using the policy based. The keys for the RSA algorithm are generated the following way

- ✓ Choose two distance prime numbers p and q.

For security purposes the integer's p and q should be chosen at random and should be of similar bit-length. Prime integers can be professionally establish using a partiality test

- ✓ Compute n = pq.
- ✓

n is used as the modules for both the public and private keys. Its span typically spoken in bits, is the key length.

- ✓ Compute φ(n) = φ(p)φ(q) = (p − 1)(q − 1), where φ is Euler's totient function.
- ✓ Choose an integer e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1; i.e. e and φ(n) are cop rime.
- e is released as the public key exponent.
- e having a short bit-length and small hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.
- ✓ Determine d as $d^{-1} \equiv e \pmod{\varphi(n)}$, i.e., d is the of e Multiplicative opposite(modulo φ(n)).
- This is more evidently stated as: solve for d known $d \cdot e \equiv 1 \pmod{\varphi(n)}$

This is often computed by means of the comprehensive Euclidean algorithms is kept as the control key exponent.

VI. RESULT

Assured deletion discuss time based deletion in and which we generalize into policy based removal. There are more than a few related systems on assured deletion which come after our conference version of the paper. Keypad protects data in theft-prone devices.
Maintaining keys in an independent centralized key server similar to FADE. It removes all statistics of a protected tool upon requests of deletion and does not consider fine grained deletion as in FADE.
Policy-based deletion follows the similar notion of ABE in which data can be accessed only if the corresponding attributes (i.e., atomic policies in our case) are satisfied.
Policy based deletion has a different design objective from ABE. Policy-based deletion focuses on how to delete data while ABE focuses on how to access data based on attributes.

A major characteristic of ABE is to subject users the decryption keys of the associated attributes so that they can access files that satisfy the attributes and hence obtainable study of ABE look for to ensure that no two users can collude if they are tied with different sets of attributes. The thought of guaranteed removal to cloud backup systems with version control, but the work does not consider access control and the use of multiple key managers for key management.

VII. CONCLUSION AND FUTURE WORK

Design and implement FADE a secure overlay cloud storage system that achieves fine-grained policy-based access control and file guaranteed removal. It acquaintances outsourced files with file access policies and assuredly deletes files to make them unrecoverable to anyone upon revocations of file access policies.
Associate files with file access policies that control how files can be access. Then present policy based file guaranteed removal in which files are assuredly deleted and made unrecoverable by anyone when their associated file access policies are revoked. We describe the essential operations on cryptographic keys so as to achieve access control and certain deletion. FADE also leverages accessible cryptographic techniques including attribute-based encryption and a quorum of key managers based on threshold secret sharing.
UIM table contains the information about the past successful file downloading results. It contains name of the service provider address resource name and count of the search. UIM table will update every possible downloading result. It also has the other sharing resource names.
It could help for future reference and it leads the dynamic search. The search based on the user's common interest. It gives the priority for the user's interest resources. It maintain the details about the meticulous resource in the memory in certain time. Key is deleted in UIM table at the certain time period users files are delete.

REFERENCES

[1] M. Nabeel and E. Bertino, "*Privacy preserving delegated access control in the storage as a service model*," in EEE International Conference on Information Reuse and Integration (IRI), 2012.
[2] E. Bertino and E. Ferrari, "*Secure and selective dissemination of XML documents*," ACM Trans. Inf. Syst. Secur., vol. 5, no. 3, pp. 290–331, 2002
[3] G. Miklau and D. Suciu, "*Controlling access to published data using cryptography*," in VLDB ‟2003: Proceedings of the 29th international conference on Very large data bases. VLDB Endowment, 2003, pp. 898–909.
[4] N. Shang, M. Nabeel, F. Paci, and E. Bertino, "*A privacy-preserving approach to policy-based content dissemination*," in ICDE ‟10: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering, 2010.
[5] M. Nabeel, E. Bertino, M. Kantarcioglu, and B. M. Thuraisingham, "*Towards privacy preserving access control in the cloud*," in Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Work sharing, ser. Collaborate Com ‟11, 2011, pp. 172–180.
[6] M.Nabeel, N.Shang,andE. Bertino, "*Privacy preserving policy based content sharing in public clouds*," IEEE Transactions on Knowledge and Data Engineering, 2012.
[7] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "*Over-encryption: Management of access control evolution on outsourced data*," in Proceedings of the

33rdInternationalConferenceonVeryLarge DataBases, ser.VLDB "07. VLDB Endowment, 2007, pp. 123–134.

[8] M. Nabeel and E. Bertino, "*Towards attribute based group key management*," in Proceedings of the 18th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2011.

[9] A.Fiat and M. Naor, "*Broadcast encryption,"* in Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, ser. CRYPTO "93. London, UK:Springer-Verlag, 1994, pp. 480–491.

[10] D. Naor, M. Naor, and J. B. Lotspiech, "*Revocation and tracing schemes for stateless receivers*," in Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, ser. CRYPTO "01. London, UK: Springer-Verlag, 2001, pp. 41–62.

[11] J. Li and N. Li, "*OACerts: Oblivious attribute certificates*," IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 4, pp. 340–352, 2006.

[12] T.Pedersen,"*Non-interactive and information-theoretic secure verifiable secret sharing*," in CRYPTO 91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology. London, UK: Springer-Verlag, 1992, pp. 129–140.

[13] M.Nabeeland E. Bertino, "*Attribute based group key management*," IEEE Transactions on Dependable and Secure Computing, 2012.

[14] A.Shamir, "*How to share a secret*," The Communication of ACM, vol. 22, pp. 612–613, November 1979.

[15] V.Shoup, "*NTL library for doing number theory*," http://www.shoup.net/ntl/.

[16] "*Open SSL the open source toolkit for SSL/TLS*," http://www.openssl.org/.

[17] "*bool stuff a boolean expression tree toolkit*,"http://sarrazip.com/dev/boolstuff.html.

[18] A. Schaad, J. Moffett, and J. Jacob, "The *role-based access control system of a european bank: a case study and discussion*," in Proceedings of the sixth ACM symposium on Access control models and technologies, ser. SACMAT "01. New York, NY, USA: ACM, 2001,

R.Manikandasamy received the B.Tech degree(first-class) in Information Technology from the Anna University Chennai 2011.currently doing the ME degree in Computer Science at Anna University Chennai/A.S.L Pauls College of Engineering & Technology ,Coimbatore .His research interest include reliability/security of cloud computing and storage, distributed systems and networks.

K.K.Kanagamathanmohan received the ME degree in computer Science at Anna University Chennai 2012.currently working assistant professor at the A.S.L Pauls College of Engineering & Technology ,Coimbatore .His research interest include reliability/security of cloud computing and storage ,distributed systems and Networks.