ACAFD: Secure and Scalable Access Control with Assured File Deletion for Outsourced Data in Cloud

Sudha Senthilkumar[1*] & Madhu Viswanatham[2]

[1]School of Information Technology and Engineering,
VIT University, Katpadi, Vellore, Tamil Nadu 632014, India
[2] School of Computer Science and Engineering,
VIT University, Katpadi, Vellore, Tamil Nadu 632014, India
*E-mail: sudha.s@vit.ac.in

Abstract. Cloud storage has emerged as a popular paradigm to outsource data to third party and share it with the collaborators. While this new paradigm enables users to outsource their sensitive data and reduces data management costs, it brings forth the new challenges to the user to keep their data secure. Data storage security and access control in the cloud is one of the challenging ongoing research works to alleviate the data leakage problem from unauthorized users. Existing solutions that use pure cryptographic techniques suffers from heavy computation work with respect to key management and key distribution. Attribute based encryption is an alternative solution that map the user access structure with the data file attributes to control the data access. However any of the existing schemes doesn't address the access control with assured deletion of the files upon revocation of user access. This article addresses this open issue using a trusted authority that manages the access control list and takes care of key management and key distribution and file deletion upon user revocation. The prototype of model has been presented and analyzed the security features over existing scheme.

Keywords: *access control; access control list; assured file deletion; cloud computing; cloud storage; data owner; trusted authority.*

1 Introduction

Cloud computing brings new level of efficiency and economy to deliver an IT resources on demand as a service over the internet. It introduces the new operational and business models for the enterprise to pay for the resources as they use, by which it relieve the enterprise burden to invest and manage their own IT infrastructure. Apart from enterprises, individuals can also use cloud services to store their large volume of data in third-party cloud servers so as to relieve the users from managing their own servers for data management. Cloud storage (Zip cloud [1], Amazon S3 [2], MyAsiaCloud [3], Googledrive [4]) offers an infinite storage space for clients to store their sensitive data. However, it imposes the data security challenges when the users or enterprises outsource their sensitive data to third-party cloud servers.

Received September 13th 2013, Revised March 13th, 2014, Accepted for publication March 28th, 2014.
 ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2014.8.1.2

As data owners move their data on untrusted cloud servers, it brings forth the high demand and concern for data confidentiality [5]. In addition to data confidentiality and privacy breach, the untrusted servers could use the data for their financial benefit and brings the huge amount of economic loss for the owners. In December 2010, first major data breach happened in Microsoft and it announced that data contained within its Business Productivity Online Suite (BPOS) has been downloaded by unauthorized users [6]. Another example is AT&T, Apple data leak protection issues in cloud breaches 100,000 of email addresses of iPad user's public [7].

There are several research work carried out to provide secure access control mechanism to protect cloud outsourced data. A direct approach is to use cryptographic techniques onto sensitive data and disclose encryption keys only to authorized users. However distributing and protecting the encryption keys from the unauthorized users create another security issue. A number of methods [8]-[10] have been recently proposed to accomplish flexible and fine-grained access control in cloud. Unfortunately these methods do not address the file deletion upon data owner request to revoke file access from the cloud user. Cloud storage provider may not totally expunge all backup file copies from its storage servers and it may disclose the data to malicious users if encryption keys are obtained by malicious attacks. Tang, *et al.* [11] addresses the file assured deletion upon revocation of user access, which focuses only the file assured deletion and do not consider the fine-grained access control.

In this paper, this open issue has been addressed for access control and file assured deletion and propose an ACAFD a secure, scalable access control with assured file deletion for cloud computing. ACAFD uses the trusted authority to decouple the access control maintenance, key management and key distribution activity from the cloud service provider. Data owner submit the access control details for the file it upload in the cloud and obtain the RSA public keys from the trusted authority. Further the data owner encrypts the file with a symmetric key and encrypts the symmetric key with session key and stores it in the Cloud Service Provider (CSP). Disclosing access details to CSP may create possibility to change the access rights when the CSP collude with any malicious users.

The remainder of the paper is organized as follows. In Section 2 related research work has been reviewed. Section 3 presents system models and assumptions. Section 4 represents algorithms of proposed scheme. In Section 5, ACAFD has been analyzed with existing scheme. Finally Section 7 concludes the paper and presents the future research direction.

2 Related Work

There are several research work carried out to address the access control issues in cloud computing. Yu, *et al.* [8] proposed a scheme to achieve secure, scalable and fine grained access control in cloud computing. They combined the techniques of Attribute-Based Encryption (ABE), Proxy re-encryption [12] and lazy re-encryption [13] to achieve the data access control on untrusted cloud-servers. Each data file can be related with attribute set. A user access structure is defined using a unique logical expression over these attributes, which indicate the user access level to access that file. Unique public keys are defined for each attributes. Data files are encrypted using the public keys associated with that attributes. Users are able to decrypt the file with secret key defined for their access structure, by which it allow the file decryption if and only if data file attribute match with the user access structure. However deriving a unique logical expression using attribute set for each data file will create heavy computation overhead for data owner. Besides, re-encryption of data file upon revocation of user access requires updating the secret keys of all user except the revoked one, becomes an issue if the number of users are large.

Wan, *et al*. [9] proposed a scheme that supports the hierarchical user structure to achieve scalable and fine-grained access control. They extend the ciphertext – policy attribute based encryption with the hierarchical user structure. Data owners who encrypt the file specify an access structure for the ciphertext. The users can decrypt the ciphertext with the decryption keys, only if the attributes specified in the key structure satisfy the access structure given by the data owner. Computation complexity varies depending upon the depth of key structure and access tree structure. Hence, the decryption is performed at the data consumer, it impact the scalability of the overall system.

Hota, *et al.* [10] proposed a capability-based access control to guarantee confidentiality, integrity and authentication in cloud outsourced data. Data owner create the capability list and update in cloud service provider. The encrypted data files and capability list are stored in cloud service provider. Based on the user request for file access, the CSP will verify the capability list and send the encrypted data file to the user. Hence the symmetric keys are known only between the data owner and data user, the CSP will not know anything about the data file. However, it may create the possibility for the CSP to change the access control information incase if it colludes with the malicious users.

Yun, *et al.* [14] proposed a cryptographic file system to provide integrity and confidentiality guarantee for outsourced data. It performs the encryption based on universal-hash based MAC tree scheme. Goh, *et al.* [15] proposed a secure

file system for remote untrusted storage called SiRiUS. It divides the file into two parts, file data and meta data file. The file data contains the encrypted and signed contents while meta data file includes the access control details. The file data encrypted using the file encryption key and signed using file signature key. The meta data file contains several encrypted key blocks that represent number of users access to that file. Each key block is encrypted with respective user master encryption key (MEK) that includes file encryption key (FEK) for read users and both file encryption key (FEK) and file signature key (FSK) for read and write privilege users. In addition, the first key block in meta data file encrypted using owner's master encryption key (MEK) and entire meta data file signed using owner's master signature key (MSK). In order to ensure freshness guarantees the timestamp is included in meta data file. However, there is no provision to ensure freshness guarantees for data file in SiRiUS that make data file are prone to replay attacks.

Tang, *et al.* [11] proposed a scheme that enables file assured deletion once the user access is revoked. The data file that is uploaded to the cloud is associated with the access policy.

The control keys are created and maintained by the key manager and it is associated to each policy. The data file will be encrypted with a data key and data key further encrypted using a control key corresponding to that file access policy. Once the access policy revoked, the corresponding control keys are deleted from the key manager. Hence, the main design objective of this scheme is to ensure file assured deletion, there is no strategy used for fine-grained access control. So our main objective in this paper is to provide the framework that incorporates both secure and scalable access control with assured file deletion.

3 System Model and Assumptions

3.1 System Model

As shown in Figure 1, the cloud computing system in our model consists of the following entities: Data owner (DO), Data Consumer (DC), Cloud Service Provider (CSP), Trusted authority (TA), Group Manager (GM), Group Member (GMem).

The DO creates the Access Control List (ACL) for the data file to be uploaded into the CSP and submit it to the TA in secure manner. In case of group access for the file, the DO submit the single ACL entry in TA to support scalability. The GM is the top-level entity that authorizes the DC who belongs to the groups. The TA is the trusted node that manages the ACL list and provides the

RSA public key based on the DO request. The DO encrypt the data file with the symmetric key and upload it in the CSP. The DC should get the certificate from the TA to access the data file resides in the CSP. Similar to [10] the DO and DC comes online only if it needs to upload and download the file in CSP. The TA, CSP and GM are always online. The DO can be a project manager in an organization who upload the project related document in a CSP or can be university teachers who upload the course material into the CSP. DC can be employees in an organization or the students who access the course materials in a university. The DO can upload the files that can be read by individual employee in an organization or it can be related to the group of employees.

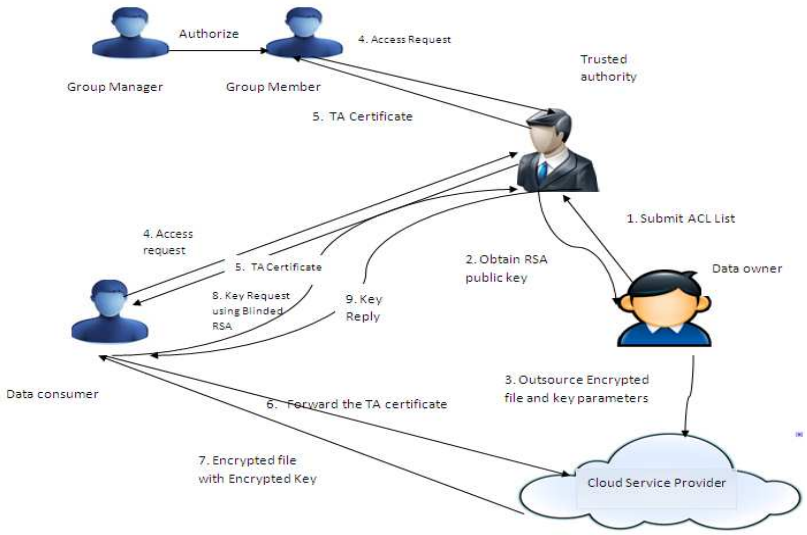

Figure 1 System Model.

3.2 Security Model

In this model, the CSP is assumed as untrusted entity that may collude with the malicious users to change the access control information, even to harm the file contents stored in the cloud. The TA is the trusted entity that resides within the DO domain to be responsible for key and access control management. TA ensures that the master key which is used to encrypt the file key will be removed in case of user revocation of file access. Also this scheme assumes that each party is already preloaded with other parties public key using an existing PKI infrastructure.

4 Algorithms of Proposed Scheme

In this section, the data structures and algorithm of ACAFD scheme has been described. Table 1 show the acronym used in our algorithm along with its definition.

Table 1 Glossary.

Acronym	Definition
K_{priv_owner}	private key of owner
K_{pub_owner}	public key of owner
K_{priv_Gmem}	private key of group member
K_{pub_Gmem}	public key of group member
K_{priv_CSP}	private key of service provider
K_{pub_CSP}	public key of service provider
f_i	i^{th} File
HMAC	Hash algorithm
FID	File ID
OwnerID	Owner identifier
AR	Access Rights
e_i, n_i	RSA public key of i^{th} user
SK	Symmetric Key for file encryption
S_i	secret key
R_i	Random number

4.1 File Upload

The DO creates the access control lists that consist of File Identifier, User Identifier, and Access Rights and File creation time and expiration time. In case of group access instead of user identifier the owner create the Access control list with the group identifier. The ACL lists are encrypted using the owner's private key first and then encrypted using TA public key and transferred to the TA. The over encryption [16] ensures the authentication and confidentiality between the DO and TA. TA decrypts the ACL List using the public key of data owner and its own private key and stores it in its database. TA generates the RSA public key e, n for the specified file id and sends it to the DO with the X.509 certificate. DO generate the random symmetric key SK for encrypting the file using the AES encryption algorithm. DO compute the hash value HMAC for the file to be uploaded and encrypt the file and HMAC using symmetric key. Along with the encrypted file, DO send the FID, OwnerID, $s_i^{ei(modni)}$ to the CSP. The creation time and expiration time in the ACL ensures the freshness of the ACL list and automatic user revocation of file access. The Figure 2 shows the file upload procedures.

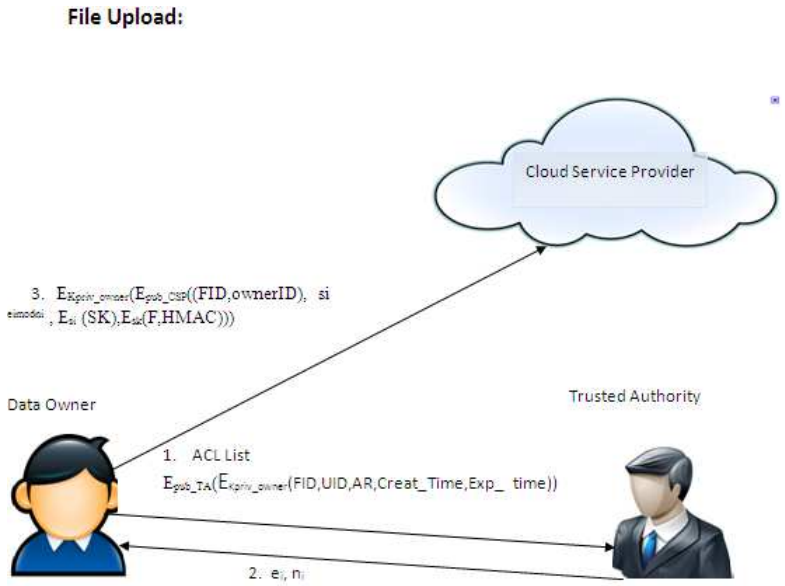

Figure 2 File uploads procedure.

Algorithm for file uploads:

Step 1: Preparing a ACL List in DO and send it to TA

DO -> TA

1.1 For each file fi create ACL

ACL <- $E_{pub\_TA}(E_{Kpriv\_owner}(FID,UID,AR,Creation\_Time, Exp\_ time))$

Step 2: Obtain RSA public key from TA

TA - > DO

Receive certificate (e_i, n_i)

Step 3: upload File to CSP

DO -> CSP

3.1 generate random symmetric key SK

3.2 encrypt the file and HMAC using SK

$E_{sk}(F,HMAC)$

3.2 Encrypt SK using s_i

E_{si} (SK)

3.3 Send the encrypted file along with FID,ownerID and key details to CSP

CSP < - $E_{Kpriv\_owner}(E_{pub\_CSP}(\ (FID,ownerID), si^{eimodni}, E_{si}$ (SK),$E_{sk}(F,HMAC)))$

3.4 CSP decrypts the file upload message and stores the files along with the file details in its database.

CSP <- $D_{Kpub\_owner}(D_{priv\_CSP}(\ (FID,ownerID), si^{eimodni}, E_{si}$ (SK),$E_{sk}(F,HMAC)))$

4.2 File Download

The DC send the request for file access to TA with the unique file id, user id and access right request along with the timestamp of request and nonce. The file access request will be encrypted using the consumer private key and TA public key for the purpose of authentication and confidentiality. The TA decrypts the request using its own private key and then with consumer public key. If DC belongs to the group, he should be authorized from GM before it can request for file access. The file access request for the group member will be encrypted using the group member private key along with (GID, AR) that is encrypted with group manager private key. The entire file request is over encrypted using TA public key for authentication and confidentiality purpose. Timestamp and Nonce in the file access request are used to eliminate the replay attacks that may occur between the communicating parties [17].

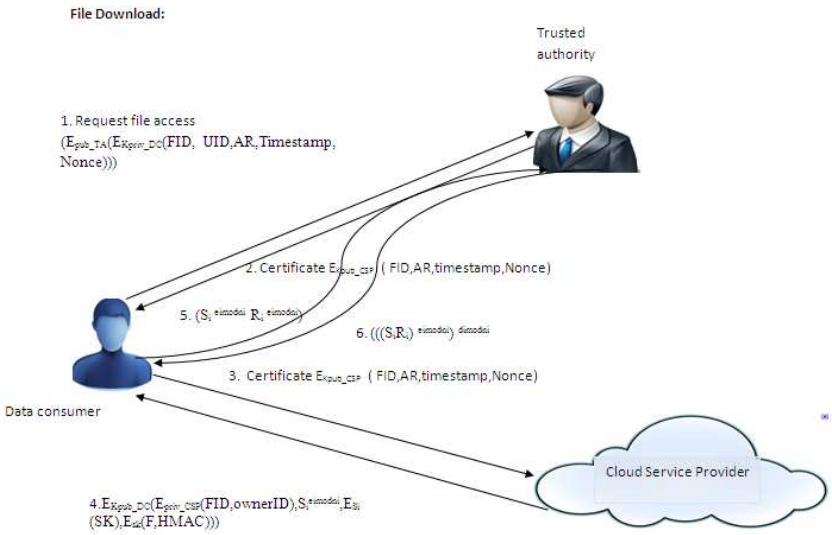

Figure 3 File Download.

Based on the file access request, TA checks ACL list to verify the authenticity of DC. In case of valid user the TA provide file access certificate to DC that are encrypted with CSP public key. The DC forwards the certificate to the CSP in order to download the file from CSP. The CSP decrypts the certificate with its private key and then fetch the file based on the certificate and send encrypted file and key details to the DC. Based on the blinded RSA [4] DC generates the random parameter R, compute $R^{eimodni}$ and include with $Si^{eimodni}$. Now DC

passes the $Si^{eimodni} R^{eimodni}$ to the TA to obtain the secret key in secure manner. The TA then computes $(S_i^{eimodni} R_i^{eimodni})^{dimodni}$ which is equal to $((SiRi)^{eimodni})^{dimodni.}$ DC will obtain the $((S_iR_i)^{eimodni})^{dimodni}$ from TA which is equal to $S_iR_{i.}$ Now DC can remove R_i from S_iR_i and obtain S_i, and decrypt Si(SK) to obtain symmetric key SK. Further DC can decrypt the file fi, HMAC using symmetric key SK [9]. The Figure 3 describes the file download procedures.

Algorithm for file downloads:

Step 1: DC or GMem send the request for file access to TA
TA <- $(E_{pub\_TA}(E_{Kpriv\_DC}(FID,UID,AR,Timestamp, Nonce)))$ or
TA <- $(E_{pub\_TA}(E_{Kpriv\_GMem}(FID,UID,AR,Timestamp, Nonce), E_{Kpriv\_GM}(GID,AR)))$
Step 2: TA provide the certificate for file access to DC
DC <- $(E_{pub\_CSP}(FID, AR,Timestamp, Nonce))$
Step 3: DC will forward the certificate to CSP
CSP <- $(E_{pub\_CSP}(FID, AR,Timestamp, Nonce))$
Step 4: CSP fetch the files based on the File ID and send to DC
DC <- $E_{Kpub\_DC}(E_{priv\_CSP}$ (FID, ownerID), $S_i^{eimodni}$, E_{Si} (SK),E_{sk}(F,HMAC)))
Step 5: DC decrypt the message and apply blind component R to secret key $S_i^{eimodni} R_i^{eimodni}$ and send to TA to request secret key
5.1 DC <- $D_{Kpriv\_DC}(D_{pub\_CSP}$ (FID, ownerID), $S_i^{eimodni}$, E_{Si} (SK),E_{sk}(F,HMAC)))
5.2 TA<- $(S_i^{eimodni} R_i^{eimodni})$
Step 6: TA compute $((S_iR_i)^{eimodni})^{dimodni}$ and pass it to DC DC <- $(((S_iR_i)^{eimodni})^{dimodni})$ => S_iR_i
Step 7: DC remove R_i from S_iR_i and obtain $S_{i,}$ decrypt symmetric key SK with $S_{i,}$ using SK it will decrypt the file and obtain HMAC and verify the integrity with newly computed HMAC.
D_{Si}(SK), D_{sk}(F,HMAC)

File Deletion: When a user leave the organization, the DO should ensure their access control is revoked from the TA. The TA delete RSA private key of corresponding file once the access control is removed from its database. Without the RSA private key $d_{i,}$ it is impossible to obtain S_i and without S_i it is not possible to know the SK to decrypt the file f_i. This meets our main design objective of assured file deletion once the user file access is revoked from CSP. In case of group user revocation, the TA keeps the RSA private key d_i until the entire users in the groups are revoked. However the user ID in the group is removed by the group manager.

5 Security Analysis

In this section, the security properties of ACAFD scheme have been analyzed and then discussed the performance analysis of ACAFD scheme in each operation.

Confidentiality and Integrity: The proposed ACAFD scheme discloses only the encrypted file details into the CSP, other details like access control information and key management are maintained by the trusted entity TA. This ensures that our scheme prevent the CSP to collude with malicious user's to change the access rights. Also the message exchange between all the communicating parties uses over encryption, which ensures the authentication and confidentiality of this scheme. Increasing key length using over encryption makes brute-force attack is very difficult. The key exchange between DO and DC are securely performed using blinded RSA method [11].

To ensure the integrity of the file, the DO compute the HMAC for the outsource file and then the file, HMAC is uploaded into CSP. After fetching a file, the DC computes the HMAC from the decrypted file and compares it with HMAC that is provided with the DO. If both match, the DC can assure the integrity of the file.

Freshness Guarantees: In order to ensure the freshness of access control information, the DO include the creation_time and expiration time Exp_time with the ACL list. In case of ACL list update for the specific file, the DO can update the creation_time and as well as expiration time. This will allow TA to perform timely revocation of user access [18] and help TA to prevent the access control rollback attacks.

Key Management: The key distributions and key maintenance are the responsibility of trusted entity TA which reduces the burden for DO to distribute the keys to authorized users. Also the TA removes the RSA private key of the files based upon the user revocation for specific file access. As per our design it is not possible to obtain the symmetric key without the RSA private key of that file. Hence the file cannot be recovered by anyone else. This implies the assured file deletion once the user access for the file is revoked.

Access Control Management: The DO is disclosing the access control details to the TA in a secured manner. It also makes a single entry for the group members to access the same file that is authorized by the GM. This supports the scalability to avoid the multiple entries for same file access. Also to make the TA more reliable, we can expand the TA with quorum of TA as mentioned in [14],[19]. As per this scheme, only files and key details have been disclosed in encrypted form which needs to be uploaded into CSP. Other than that the CSP will not know any information related to access details of the files. This prevents the CSP to collude with malicious users to change the access control information [8].

6 Performance Analysis

New File Creation: To create a new file, the DO needs to update the ACL entry in TA, and should obtain RSA public key information from the TA. After this DO can generate the random symmetric key by which they can encrypt the data file. Further, symmetric key are encrypted using a secret key and secret key are raised to the power $e_i mod n_i$ to transmit key securely. Here the complexity of file creation involves size of data file, the type of symmetric key algorithm used, algorithm used for encrypting a symmetric key and secret key raised to the power of $e_i mod n_i$. Assume T1 is a total time required to encrypt a file f_i, encrypt symmetric key SK using S_i and raise a secret key to the power $e_i mod n_i$. T2 is the transmission time to upload the file in CSP and T3 is the decryption time at CSP then time complexity for new file creation is O(N) where N = T1+T2+T3.

Efficient File Access: DC can request for file access to the TA. TA verifies the authority of the DC and if DC is valid user, it prepares the certificate and sends it to DC. DC forwards the certificate to CSP to obtain the file information. According to the TA certificate, CSP obtain the file details from its database and transmit to DC. Further DC request TA to obtain secret key using blinded RSA and then decrypt the files. Assume if T1 is the total time required to prepare file access request and transmit, time required to receive the certificate and time required to forward the certificate. T2 is the file transmission time from CSP to TA and T3 is the request for secret key and reply and T4 is the decryption time at DC then time complexity is O(N) where N = T1+T2+T3+T4.

Efficient User Revocation: To deal with efficient user revocation, we have included the expiration time in each ACL entry by which the TA can perform automatic user revocation. Once the DO request the removal of use access rights, the TA remove the specific ACL entry from its database. So the time complexity to revoke the use access is O(1).

File Deletion: After deletion of ACL entry, the TA deletes the RSA private key of that user associated with that file. Without the RSA private key, the data file in CSP is no more accessible by any users. This fulfills the file assured deletion of our ACAFD scheme. The time complexity for file deletion is O(1).

7 Conclusion and Future Work

In this paper ACAFD prototype has been presented, which provides the secure access control with assured file deletion. This scheme seamlessly integrates set of cryptographic file operations to secure the files outsourced to cloud. Further this scheme empowers the DO to upload the file into CSP without disclosing the access control information in CSP. This scheme not only provides the efficient

access to the file, but also achieves the efficient user revocation, file deletion and freshness guarantees in access control. The theoretical security and performance analyzes have been discussed which provides the insight into security–performance trade-off of this scheme when it is deployed into cloud.

In future extension, this ACAFD scheme will be implemented into real public cloud like Amazon S3 or Microsoft Azure to prove the security of this scheme. Further the quorum scheme into the TA will be implemented to enhance the reliability of the TA in case of access control and key management. Enhancement into this current design involves including the trusted third party auditor service to assess the cloud service risks.

References

[1] ZipCloud, http://www.zipcloud.com (1 November 2013).
[2] Amazon Simple Storage Service, http://aws.amazon.com/s3 (1 November 2013).
[3] MyAsiaCloud, http://www.myasiacloud.com/ (1 September 2013).
[4] GoogleDrive, https://drive.google.com (1 September 2013).
[5] di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S. & Samarati, P., *A Data Outsourcing Architecture Combining Cryptography and Access Control*, Proc. ACM Workshop on Computer Security Architecture (CSAW'07), USA, Nov 2007.
[6] Thomas, K., *Microsoft Cloud Data Breach Heralds Things to Come*, PC World, http://www.pcworld.com/article/214775/microsoft_cloud_data_breach_sign_of_future.html (1 November 2013).
[7] Deltcheva, R., *Apple, AT&T Data Leak Protection Issues Latest in Cloud Failures*, http://www.messagingarchitects.com/resources/security-compliance-news/email-security/apple-att-data-leak-protection-issues-latest-in-cloud-failures19836720.html (1 November 2013).
[8] Yu, S., Wang, C., Ren, K. & Lou, W., *Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing*, in Proc. IEEE INFOCOM 2010, San Diego, CA, pp. 534-542, 2010.
[9] Wan, Z., Liu, J. & Deng, R.H., *HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing*, In Proc. IEEE Transcations on Information Forensics and Security, **7**(2), April 2012.
[10] Hota, C., Sanka, S., Rajarajan, M. & Nair, S.K., *Capability-based Cryptographic Data Access Control in Cloud Computing*, International Journal of Advanced Networking and Applications, **3**(3), pp. 1152-1161, 2011.

[11] Tang, Y., Lee, P.P.C, Lui, J.C.S. & Perlman, R., *FADE: Secure Overlay Cloud Storage with File Assured Deletion*, IEEE Transcations Dependable on Secure Computing, **9**(6), 2012.

[12] Blaze, M., Bleumer, G. & Strauss, M., *Divertible Protocols and Atomic Proxy Cryptography*, in Proc. of EUROCRYPT '98, 1998.

[13] Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q. & Fu, K., *Scalable Secure File Sharing on Untrusted Storage*, in Proc. of FAST '03, 2003.

[14] Yun, A., Shi, C. & Kim, Y., *On Protecting Integrity and Confidentiality of Cryptographic File System for Outsourced Storage*, In ACM Cloud Computing Security Workshop (CCSW), Chicago, Illinois, USA, Nov. 2009

[15] Goh, E., Shacham, H., Modadugu, N. & Boneh, D., *Sirius: Securing Remote Untrusted Storage*, Proc. Network and Distributed Systems Security Symposium (NDSS'03), San Diego, California, USA, pp. 131-145, 2003

[16] di Vimercati, S.D.C., Foresti, S., Jajodia, S., Samarati, P. & Paraboschi, S., *Over-encryption: Management of Access Control Evoluation on Outsourced Data*, Proc. 33rd International Conference on Very Large Databases (VLDB'07), Vienna, Austria, pp. 123-134, 2007

[17] William Stallings, *Cryptography and Network Security*, Prentice Hall Upper Saddle River, N.J., 2006.

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

[19] Shamir, A., *How to Share a Secret*, Communication of the ACM, **22**(11), pp. 612-613, Nov 1979.