RESEARCH ARTICLE OPEN ACCESS

Cloud audit service outsourcing by using KAC for data storage security

Ms.A.Christy Sharmila
PG Scholar
Department of CSE
Loyola Inst. of Tech. & Science
Thovalai-629 302.
[1]chrstshrml@gmail.com

Mr.T.Darney Tressiline M.E.,
Assistant Professor
Department of CSE
Loyola Inst. of Tech. & Science
Thovalai-629 302.
[2]darneytressiline@rocketmail.com

Abstract:
Cloud computing is recognized as an alternative to traditional information technology due to its intrinsic resource-sharing and low-maintenance characteristics. Enterprises usually store data in internal storage and install firewalls to protect against intruders to access the data. With proven security relied on number theoretic assumptions is more desirable, whenever the user is not perfectly happy with trusting the security of the Virtual Machine or the honesty of the technical staff. The challenging problem is how to effectively share encrypted data. The proposed method is a new technique named Key Aggregate Cryptosystem(KAC), which is based on tree structure in providing keys to the files. Each leaf node file has a secret key for decrypting that ciphertext class of that file. And the parent node also have a secret key. If the key is granted to receiver for the leaf node, that particular file only decrypted. And if the key is granted for the parent node, then the receiver have rights to decrypt the parent node and the leaf nodes which are under that parent node. The remaining node remains same. The objective of this scheme is to design an efficient public-key encryption scheme which supports flexible delegation in the sense that any subset of the ciphertexts (produced by the encryption scheme) is decryptable by a constant-size decryption key (generated by the owner of the master-secret key). Through this, the user can share more files with a single key at a time.

Index Terms: encryption, parent node, leaf node, ciphertext

I. INTRODUCTION

In recent years, the emerging cloud-computing is rapidly gaining thrust as an alternative to traditional computing system. Cloud computing provides a scalability environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-services. But By seeing the popularities of cloud computing services, it's fast development and advance technologies anyone can voted it as a future of computing world. Cloud stores the information that is locally stores in the computer, it contains spreadsheets, presentations, audio, photos, word processing documents, videos, records, photos. But for sensitive and confidential data there should be some security mechanism, so as to provide protection for private data [19] [20].

Conventionally, client can verify the data integrity based on two-party storage auditing protocols [4] [5] [6]. But it is inefficient for auditing, because no one can give (i.e. client or cloud service provider) assurance to provide balance auditing.

Therefore third-party auditing (TPA) is play important role for the storage auditing in cloud computing. It is very convenient for both side i.e. owner side and cloud service provider side.

One fundamental aspect of this paradigm shifting is that data are being centralized and outsourced into clouds. This kind of outsourced storage services in clouds have become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data.

The cloud storage service (CSS) relieves the burden of storage management and maintenance. However, if such an important service is susceptible to attacks or failures, it would bring irretrievable losses to users since their data or archives are stored into an uncertain storage pool outside the enterprises. These security risks come from the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices. However, they are still susceptible to security threats both from outside and inside the cloud for the benefits of their possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users furthermore, the dispute occasionally suffers from the lack of trust on CSP [1].

Consequently, their behaviors may not be known by the cloud users, even if this dispute may result from the users' own improper operations. Therefore, it is necessary for cloud service providers to offer an efficient audit service to check the integrity and availability of the stored data. Traditional cryptographic technologies for data integrity and availability, based on hash functions and signature scheme, cannot work on the outsourced data without a local copy of data.

In addition, it is not a practical solution for data validation by downloading them due to the expensive transaction, especially for large-size files. Moreover, the solutions to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users [1] [2] [3]. Therefore, it is crucial to realize public audit ability for CSS, so that data owners may resort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data.

II. EXISTING SYSTEM AND CHALLENGES

Ateniese et al. [7] are the first to consider public auditability in their "provable data possession" (PDP) model for ensuring possession of data files on untrusted storages. They utilize the

RSA-based homomorphic linear authenticators for auditing outsourced data and suggest randomly sampling a few blocks of the file. However, among their two proposed schemes, the one with public auditability exposes the linear combination of sampled blocks to external auditor. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the external auditor.

Juels et al. [9] describe a "proof of retrievability" (PoR) model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on remote archive service systems. However, the number of audit challenges a user can perform is fixed a priori, and public auditability is not supported in their main scheme. The authors complete their dynamic auditing system to be privacy preserving and it support the batch auditing for multiple owners [11]. However, due to the large number of data tags, their auditing protocols will incur a heavy storage overhead on the server.

In [10], the authors proposed a dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers, but this method may leak the data content to the auditor because it requires the server to send the linear combinations of data blocks to the auditor.

In Yan Zhu used [1] a quantified new audit approach based on probabilistic queries and periodic verification, as well as an optimization method of parameters of cloud audit services. This approach greatly reduces the workload on the storage servers, while still achieves the detection of servers' misbehavior with a high probability.

By referring different existing system, we have described some suggested requirements for public auditing services and the state of threat that fulfills them. However, this is still not enough for a publicly auditable secure cloud data storage system, and further challenging issues remain to be supported and resolved.

(1) What will happen if the data owner and TPA are unreliable? In this case the auditing result should identify the data correctness as well as it should be able to determine which entity is responsible for the problem like owner, TPA or cloud server. So systems accountability should be achieved.

(2) Performance is another important aspect in cloud computing data storage security and its integrity for any physical system.

(3) Cloud data storage provides dynamic and scalable storage services. It also allows easy on-demand file sharing on cloud. The challenge in this case is that legacy users, who access data and it can also modify the owner's data in the cloud. So major challenge is dynamics support for public auditing services while maintaining system runtime.

To securely launch an effective third party auditor (TPA), the following two essential requirements should be met;

1) TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user.

2) The third party auditing process should bring in no new vulnerabilities towards user data privacy. Therefore our system protocol provide Confidentiality, Dynamic auditing and Batch auditing i.e. auditing protocol is able to support the batch auditing for many owners and many clouds[12][13].

III. AUDIT SYSTEM ARCHITECTURE

The audit system architecture for outsourced data in clouds in which can work in an audit service outsourcing approach. In this architecture, we reflect on a data storage service containing four entities:

1) Data owner (DO): who has data files to be stored in the cloud and relies on the cloud for data maintenance, can be an individual customer or an organization.

2) Cloud Storage Service Provider (CSP): who provides data storage service and has enough storage space to maintain client's data.

3) Third Party Auditor (TPA): a trusted person who manage or monitor outsourced data under request of the data owner.

4) Authorized Application (AA): who have the right to access and manipulate stored data.

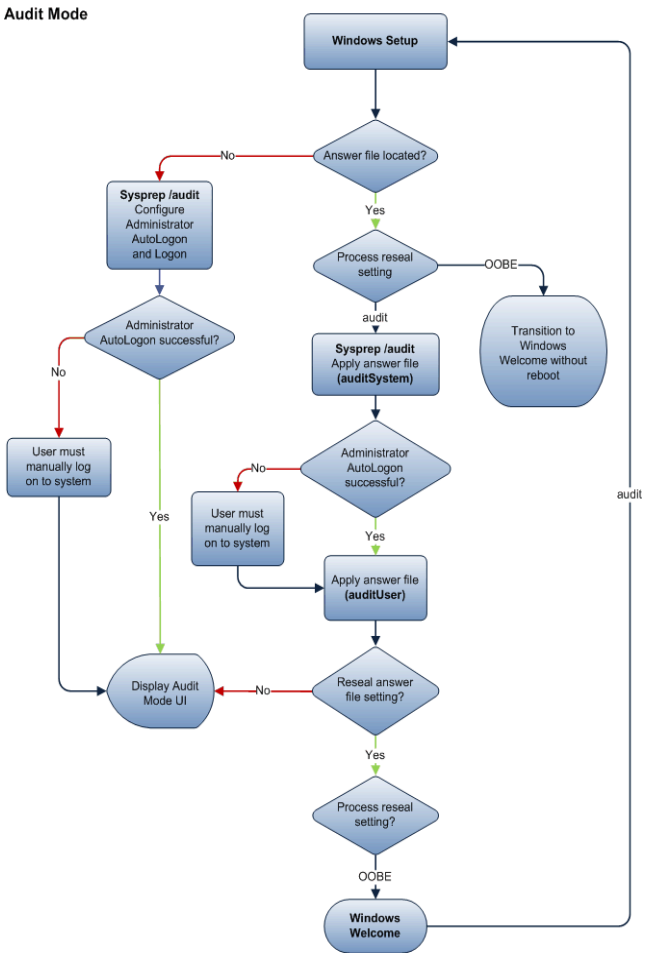

Fig.1.Flow chart for Audit service

The data which the data owner wants to store in cloud first reaches the authorized application which will create digital signature and sends the data to the cloud storage. If the user needs to verify data means the verification request should be send to third party auditor (TPA), the TPA will retrieve the digital signature from the database and will send the verification request to the management server. The management server in turn will generate the digital signature for the data stored in the cloud and it will send only that digital signature instead of the whole data to the TPA. The TPA will decrypt the digital signature and compares the message digest for verifying correctness of data.

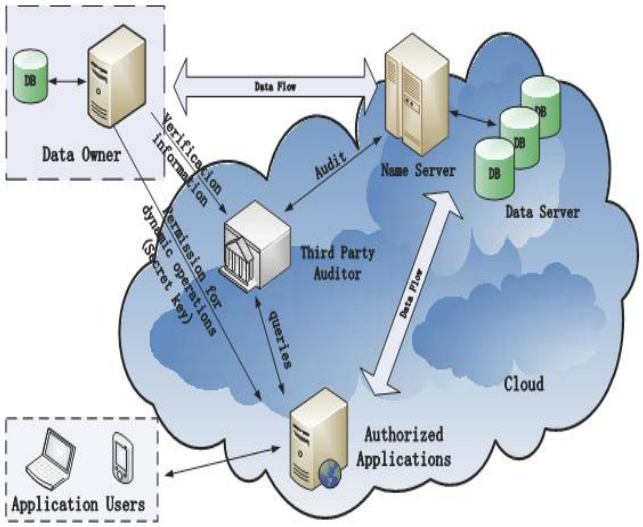

Fig.2.Architecture Diagram

This architecture is known as the audit service outsourcing due to data integrity authentication. Architecture contains the data owner and granted clients need to dynamically interact with cloud service provider to access or update their data for various application purposes. However, we neither assume that cloud service provider is trust to guarantee the security of stored data, or suppose that the data owner has the capability to collect the verifications of cloud service provider's fault after errors occur. Hence, third party auditor, as a trust third party (TTP), is used to ensure the storage security of their outsourced data. We assume the third party auditor is reliable and independent, and thus has no encouragement to join together with either the cloud service provider or the clients during the auditing process:

• TPA must be able to make regular check on the integrity and availability of these delegated data at appropriate intervals;

• TPA must be able to take the evidences for the disputes about the inconsistency of data in terms of authentic records for all data operations.

To facilitate privacy-preserving public auditing for cloud data storage beneath the architecture, the protocol design should attain subsequent security and performance guarantees:

1) Audit-without-downloading: to allow TPA (or other clients with the help of TPA) to authenticate the correctness of cloud data on demand without recovering a copy of whole data or bring in additional on-line burden to the cloud users;

2) Verification-correctness: to make sure there exists no unethical CSP that can pass the audit from TPA without indeed storing users' data intact;

3) Privacy-preserving: to make sure that there exists no way for TPA to derive users' data from the in sequence collected during the auditing process;

4) High-performance: to allow TPA to perform auditing with minimum overheads in storage, communication and computation, and to maintain statistical audit sampling and optimized audit schedule with a long enough period of time.

IV. KEY-AGGREGATE ENCRYPTION

The above processes involve some procedures: TagGen, KeyGen, Update, Insert, Delete, algorithms as well as an interactive proof protocol of retrievability. In order to improve security and performance, we make use of following techniques to construct corresponding algorithms and protocols.

KeyGen(1^κ)**:** Let $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear map group system with randomly selected generators $g, h \in_R \mathbb{G}$, where $\mathbb{G}$, $\mathbb{G}_T$ are two group of large prime order p, $|p| = O(\kappa)$. Generate a collision-resistant hash function $H_k(\cdot)$ and chooses a random $\alpha, \beta \in_R \mathbb{Z}_p$ and computes $H_1 = h^\alpha$ and $H_2 = h^\beta \in \mathbb{G}$. Thus, the secret key is $sk = (\alpha, \beta)$ and the public key is $pk = (g, h, H_1, H_2)$.

TagGen(sk, F)**:** Splits the file F into $n \times s$ sectors $F = \{m_{i,j}\} \in \mathbb{Z}_p^{n \times s}$. Chooses s random $\tau_1, \cdots, \tau_s \in \mathbb{Z}_p$ as the secret of this file and computes $u_i = g^{\tau_i} \in \mathbb{G}$ for $i \in [1, s]$ and $\xi^{(1)} = H_\xi(\text{"}Fn\text{"})$, where $\xi = \sum_{i=1}^{s} \tau_i$ and Fn is the file name. Builds an index table $\chi = \{\chi_i\}_{i=1}^{n}$, then calculates its tag as

$$\sigma_i \leftarrow (\xi_i^{(2)})^\alpha \cdot g^{\sum_{j=1}^{s} \tau_j \cdot m_{i,j} \cdot \beta} \in \mathbb{G}.$$

where $\xi_i^{(2)} = H_{\xi^{(1)}}(\chi_i)$ and $i \in [1, n]$. Finally, sets $u = (\xi^{(1)}, u_1, \cdots, u_s)$ and outputs $\zeta = (\tau_1, \cdots, \tau_s)$, $\psi = (u, \chi)$ to TTP, and $\sigma = (\sigma_1, \cdots, \sigma_n)$ to CSP.

Proof(CSP, TPA)**:** This is a 3-move protocol between CSP and TPA with the common input (pk, ψ), as follows:

- **Commitment**$(CSP \rightarrow TPA)$: CSP chooses a random $\gamma \in \mathbb{Z}_p$ and s random $\lambda_j \in_R \mathbb{Z}_p$ for $j \in [1, s]$, and sends its commitment $C = (H_1', \pi)$ to TPA, where $H_1' = H_1^\gamma$ and $\pi \leftarrow e(\prod_{j=1}^{s} u_j^{\lambda_j}, H_2)$;
- **Challenge**$(CSP \leftarrow TPA)$: TPA chooses a random challenge set I of t indexes along with t random coefficients $v_i \in \mathbb{Z}_p$. Let Q be the set $\{(i, v_i)\}_{i \in I}$ of challenge index coefficient pairs. TPA sends Q to CSP;
- **Response**$(CSP \rightarrow TPA)$: CSP calculates the response θ, μ as

$$\begin{cases} \sigma' \leftarrow \prod_{(i,v_i) \in Q} \sigma_i^{\gamma \cdot v_i}, \\ \mu_j \leftarrow \lambda_j + \gamma \cdot \sum_{(i,v_i) \in Q} v_i \cdot m_{i,j}, \end{cases}$$

where $\mu = \{\mu_j\}_{j \in [1,s]}$. P sends $\theta = (\sigma', \mu)$ to V;

Verification: TPA can check that the response was correctly formed by checking that

$$\pi \cdot e(\sigma', h) \stackrel{?}{=} e(\prod_{(i,v_i) \in Q} (\xi_i^{(2)})^{v_i}, H_1') \cdot e(\prod_{j=1}^{s} u_j^{\mu_j}, H_2).$$

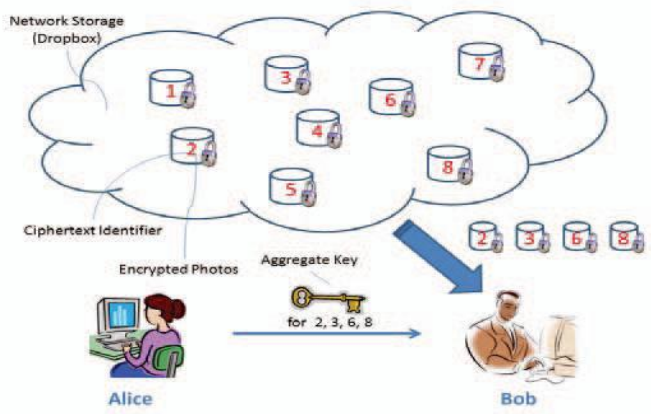

Fig.3.Alice shares files with identifiers 2, 3, 6 and 8 with Bob by sending him a single aggregate key.

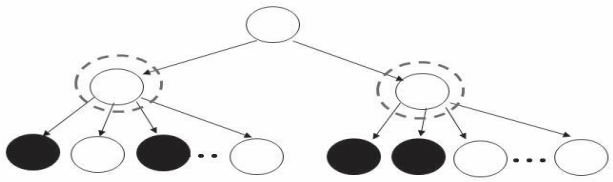

Fig.4.Key Assignment

V. CRYPTOSYSTEM ME6

Plaintext is readable data (for example, a spreadsheet file), and ciphertext is the result of encrypting plaintext. A cryptosystem is a set of procedures and conventions for hiding and revealing information in a controlled manner. A cryptosystem generally has two distinct components: (a) the processes used to encipher and decipher data and (b) the set of keys used to influence the operation of these processes so that the ciphertext is dependent on the key used for encryption. The security of a cryptosystem lies not in the secrecy of the methods used to encipher and decipher the data but rather in the difficulty of decrypting ciphertext in the absence of knowledge of the key used to produce it.

Cryptosystem ME6 encrypts data in files stored on disk. A file may be considered as a sequence of at least one byte and perhaps many millions of bytes. ME6 reads in plaintext from a file in blocks whose size is between 6 KB and 10 KB (the exact size of each block depends on the encryption key), encrypts each block and writes the resulting ciphertext to disk. This is done for each of the blocks making up the file. Each block is first compressed, if possible, before being encrypted, so normally the ciphertext blocks are smaller than the plaintext blocks, with the result that the file containing the encrypted data is usually smaller than the input file.

VI. THE ENCRYPTION METHOD

A file to be encrypted or decrypted is treated as a series of
blocks (6 KB to 10 KB in size) and the bytes within each block
are treated as an input stream of bytes. The algorithm is an auto
key stream cipher with the key text generation dependent on the foregoing plaintext (making use of the MD5 message digest algorithm).The key text stream is made up of segments. The first, the "priming key", is formed from the encryption key. The length of the priming key and the lengths of the subsequent key text segments are not constant but depend on the encryption key. Two pseudo random number generators are used (a linear congruential generator and a Park-Miller generator).

The plaintext stream (the contents of the file being encrypted) is divided into segments of lengths equal to the key text segments. A ciphertext stream is produced which consists of segments of lengths equal to the plaintext and key text segments. The encrypted file consists of the entire ciphertext stream. The input stream is read in a series of blocks which can be any size in the range 6 KB to 10 KB, the exact sizes being dependent on the encryption key. Two encryption processes are used, an "inner" and an "outer" process. The outer process is applied to the blocks. Within each block the inner process is applied to sub-blocks which are 200 to 400 bytes in size (the plaintext segments).

VII. THE DECRYPTION METHOD

The decryption process undoes what the encryption process does. A priming key is formed in the same way as in the encryption process and a key text stream is generated which is identical to the key text stream used during encryption. Each block of the ciphertext stream written during encryption is read and its constituent ciphertext segments are subjected to the inner decryption process, as described below. The outer decryption process is then applied to the block. This is repeated for each block of ciphertext to recover the original file.

For the inner decryption process, first the bytes in each ciphertext segment are unshuffled. For i >= 0 each plaintext segment p(i) = k(i) XOR c(i), where c(i) is the i-th ciphertext segment within the block and k(0) and k(i) for i > 0 are as defined above for the inner encryption process. For the outer decryption process, the combination with the output of the random number generators is undone. The block is decompressed and the two halves are interchanged to recover the original plaintext block.

VIII. IMPLEMENTATION AND RESULTS

To authorize the effectiveness of our approach, we have implemented a prototype of an audit system based on our proposed solution. This system have been developed in an experimental cloud computing system environment, which is constructed within the framework of the IaaS to provide powerful virtualization, distributed storage, and automated management. How to protect users' data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In this article, we consider how to "compress" secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegatee can always get an aggregate key of constant size. The files are stored in cloud (other parties control), also he can't able to access the file. All files are gets encrypted. The third party can't able to provide keys to unauthorised users privately without the permission of owner. By our evaluation, this method gives high percent of secure than the existing system.

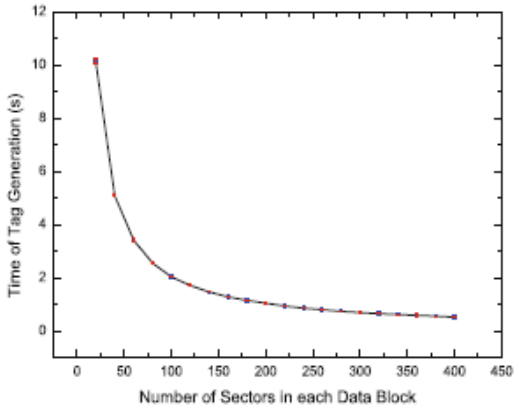

Fig.5.Computation time of tag generation for 1MByte data

IX. CONCLUSION

Cloud Computing releases the world of computing to a wider range of uses and increases the ease of usage by giving access through any kind of internet connection. Though with these increased ease of usage also come drawbacks. Privacy security is a key issue for cloud storage and is to be considered very important. To ensure that the risks of privacy have been mitigated a variety of techniques that may be used in order to achieve privacy. This paper has addressed some privacy approaches for overcoming the issues in privacy on un trusted data stores in cloud computing. Categories the methodologies in the literature as encryption

based methods, access control based mechanisms, query integrity/ keyword search schemes, and audit ability schemes. The work is giving an efficient privacy-preserving storage compared to other works. Even though there are many approaches in the literature for mitigating the concerns in privacy, no approach is fully sophisticated to give a privacy-preserving storage that overcomes all the other privacy concerns. Thus to deal with the concerns of privacy, we need to develop privacy–preserving framework that overcomes the worries in privacy security and encourage users to adopt cloud storage services more confidently.

Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of ciphertexts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension.

REFERENCES

[1] Yan Zhua,b, Hongxin Huc, Gail-Joon Ahnc, Stephen S. Yauc. "Efficient audit service outsourcing for data integrity in clouds". In "The Journal of Systems and Software 85 (2012) 1083– 1095".

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A.Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M.Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, 2010.

[3] T. Velte, A. Velte, and R. Elsenpeter, Cloud Computing: A Practical Approach, first ed., ch. 7. McGraw-Hill, 2010.

[4] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 598-609, Oct. 2007.

[6] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6,2007.

[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 598-609, 2007.

[8] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[9] A. Juels and J. Burton, S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[10] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel Distributed Systems, vol. 22, no. 5,pp. 847-859, May 2011.

[11] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533, 2010.

[12] C. Wang, K. Ren, W. Lou, and J. Li, "Toward Publicly Auditable Secure Cloud Data Storage Services," IEEE Network, vol. 24, no. 4, pp. 19-24, July/Aug. 2010.

[13] K. Yang and X. Jia, "Data Storage Auditing Service in Cloud Computing: Challenges, Methods and Opportunities," World Wide Web, vol. 15, no. 4, pp. 409-428, 2012.

[14] Q. Wang et al., "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. ESORICS '09, Sept. 2009, pp. 355–70.

[15] C. Erway et al., "Dynamic Provable Data Possession," Proc. ACM CCS '09, Nov. 2009, pp. 213–222.

[16] C. Wang et al., "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc. IEEE INFOCOM '10, Mar. 2010.

[17] Cong Wang and Kui Ren, Illinois Institute of Technology Wenjing Lou, Worcester Polytechnic Institute Jin Li, Illinois Institute of Technology "Toward Publicly Auditable Secure Cloud Data Storage Services". 0890-8044/10/2010 IEEE.

[18] Cong Wang, Student Member, IEEE, Qian Wang, Student Member, IEEE, Kui Ren, Senior Member, IEEE, Ning Cao, and Wenjing Lou, Senior Member, IEEE "Toward Secure and Dependable Storage Services in Cloud Computing" IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 5, NO. 2, APRIL-JUNE 2012.

[19] Kan Yang, Student Member, IEEE, and Xiaohua Jia, Fellow, IEEE "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing" IEEE

TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 9, SEPTEMBER 2013.

[20] Cong Wang, Member, IEEE, Sherman S.M. Chow, Qian Wang, Member, IEEE, Kui Ren, Senior Member, IEEE, and Wenjing Lou, Senior Member, IEEE "Privacy-Preserving Public Auditing for Secure Cloud Storage" IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 2, FEBRUARY 2013.

[21] Cheng-Kang Chu, Sherman S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng, Senior Member, IEEE," Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage," IEEE Transactions on Parallel and Distributed Systems,IEEE,2013.

[22] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Trans.Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[23] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in *International Conference on Distributed Computing Systems - ICDCS 2013*. IEEE, 2013.

[24] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken,"Dynamic and Efficient Key Management for Access Hierarchies," *ACMTransactions on Information and System Security (TISSEC)*, vol. 12,no. 3, 2009.

[25] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 14, no. 1, pp. 182–188,2002.