IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-3, Issue-5 E-ISSN: 2347-2693

Privacy Preserving Collaborative Auditing Data Storage Scheme in Cloud Computing

Bullarao Domathoti[1*], Rajia Begum[2] and Nageswara Rao.P[3]

[1,2,3]*Department of Computer Science and Engineering , SITS JNTUA, INDIA,*

www.ijcseonline.org

Received: May/02/2015 Revised: May/07/2015 Accepted: May/25/2015 Published: May/31/ 2015

Abstract— Cloud services provide great conveniences for the users to enjoy the on-demand cloud applications without considering the local infrastructure limitations. During the data accessing, different users may be in a collaborative relationship, and thus data sharing becomes significant to achieve productive benefits. The existing security solutions mainly focus on the authentication to realize that a user's privative data cannot be unauthorized accessed, but neglect a subtle privacy issue during a user challenging the cloud server to request other users for data sharing. The challenged access request itself may reveal the user's privacy no matter whether or not it can obtain the data access permissions. Several schemes employing attribute-based encryption (ABE) have been proposed for access control of outsourced data in cloud computing. Thus, enabling public auditability for cloud data storage security is of critical importance so that users can resort to an external audit party to check the integrity of outsourced data when needed. To securely introduce an effective third party auditor (TPA), the following two fundamental requirements have to 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. In this paper, we utilize the public key based homomorphic authenticator and uniquely integrate it with random mask technique to achieve a privacy-preserving public auditing system for cloud data storage security while keeping all above requirements in mind. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient

Keywords— Cloud Computing, Authentication Protocol, Privacy Preservation, Shared Authority, Universal Composability

I. INTRODUCTION

CLOUD computing is a promising information technology architecture for both enterprises and individuals. It launches an attractive data storage andinteractive paradigm with obvious advantages, including on-demand self-services, ubiquitous network access,and location independent resource pooling [1]. As a disruptive technology with profound implications, Cloud Computing is transforming the very nature of how businesses use information technology. Subsequently, security and privacy issues are becoming key concerns with the increasing popularity of cloud services. [2]Conventional security approaches mainly focus on the strong authentication to realize that a user can remotely access its own data in on-demand mode. One fundamental aspect of this paradigm shifting is that data is being centralized or outsourced into the Cloud. From users' perspective, including both individuals and IT enterprises, storing data remotely into the cloud in a flexible on-demand manner brings appealing benefits: relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc [2]. although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity. Examples of outages and security breaches of noteworthy cloud services appear from time to time [3–6]. Secondly, for the benefits of their own, there do exist various motivations for cloud service providers to behave unfaithfully2 towards the cloud users regarding the status of their outsourced data. Examples include cloud service providers, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed, or even hiding data loss incidents so as to maintain a reputation [7–9]. In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, it does not offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the successful deployment of the cloud architecture. Along with the diversity of the application requirements, users may want to access and share each other's authorized data fields to achieve pro-ductive benefits, which brings new security and privacy challenges for the cloud storage.

An example to identify the management storage data like demanding and suppliers , demand of products, type of security data, market basket data, drop box data etc.

In market storage data available products, selling price, quantity, suppliers. This type of data we share in cloud data storage schema was minting. the database schema divide into groups. Each group owns its users which are permitted to access the authorized data fields, and different users own relatively independent access authorities. It means that any more users from diverse groups should access different data fields of the same file. The files using clouds are unarguable, due to the opaqueness of the Cloud—as separate administrative entities, the internal operation details of cloud service providers (CSP) may not be known by cloud users—data outsourcing is also relinquishing user's ultimate control over the fate of their data. As a result, the correctness of the data in the cloud is being put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity. Examples of outages and security breaches of noteworthy cloud services appear from time to time [3–6]. Secondly, for the benefits of their own, there do exist various motivations for cloud service providers to behave unfaithfully towards the cloud users regarding the status of their outsourced data. Examples include cloud service providers, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed, or even hiding data loss incidents so as to maintain a reputation [7–9]. In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, it does not offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the successful deployment of the cloud architecture.

As users no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted. Thus, how to efficiently verify the correctness of outsourced cloud data without the local copy of data files becomes a big challenge for data storage security in Cloud Computing. Note that simply downloading the data for its integrity verification is not a practical solution due to the expensiveness in I/O cost and transmitting the file across the network. Besides, it is often insufficient to detect the data corruption when accessing the data, as it might be too late for recover the data loss or damage. Considering the large size of the outsourced data and the user's constrained resource capability, the ability to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users [9,10].

For the third party auditing in cloud storage systems, there are several important requirements which have been proposed in some previous works [18], [19]. Auditing protocol should have the following properties: 1)Authentication: a legal user can access its own data fields,only the authorized partial or entire data fields can be identified by the legal user, and any forged or tampered data fields cannot deceive the legal user. 2) Confidentiality.The auditing protocol should keep owner's data confidential against the auditor. 3) Dynamic Auditing. The auditing protocol should support the dynamic updates of the data in the cloud. 4) Batch Auditing. The auditing protocol should also be able to support the batch auditing for multiple owners and multiple clouds. 5) Forward security: any adversary cannot correlate two communication sessions to derive the prior interrogations according to the currently captured messages.

Researches have been worked to strengthen security protection and privacy preservation in cloud applications, and there are various cryptographic algorithms to address potential security and privacy problems, including security architectures [4], [5], data possession protocols [6], [7], data public auditing protocols [8]–[10], secure data storage and data sharing protocols [11]–[16], access control mechanisms [17]–[19], privacy preserving protocols [20]–[23], and key management [24]–[27]. 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 challenges the cloud server to request other users for data sharing, the access request itself may reveal the user's privacy no matter whether or not it can obtain the data access permissions. we aim to address a user's sensitive access desire related privacy during data sharing in the cloud environments, and it is significant to design a humanistic security scheme to simultaneously achieve data access control, access authority sharing, and privacy preservation. Specifically, our contribution in this work can be summarized as the following three aspects:

1) We motivate the public auditing system of data storage security in Cloud Computing and provide a privacy-preserving auditing protocol, i.e., our scheme supports an external auditor to audit user's outsourced data in the cloud without learning knowledge on the data content.

2) To the best of our knowledge, our scheme is the first to support scalable and efficient public auditing in the Cloud Computing. In particular, our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA.

3) We prove the security and justify the performance of our proposed schemes through concrete experiments and comparisons with the state-of-the-art.

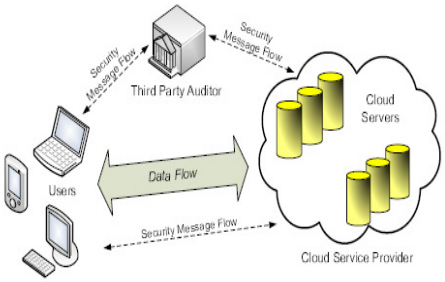


Fig 1TPA security service provider by cloud server

II. LITERATURE SURVEY:

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things r satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

PROBLEM STATEMENT: In our model, privacy is accomplished by encrypting the data it can prevent the un authorized access. We are going to raise the privacy level of the data owner and the confidentiality of the data by providing access to users above Architecture

1. Owner registration: In this an owner has to upload its files in a cloud server, he/she should register first. Then only he/she can be able to do it. For that he needs to fill the details in the registration form. These details are maintained in a database.

2. Owner: In this module,any of the above mentioned person have to login,they should login by giving their emailid and password .

2. cloud service provider : system model and security definition are presented in this section. An ID-DPDP protocol comprises four different entities which are illustrated in Figure 1. We describe them below:
1) Client: an entity, which has massive data to be stored on the multi-cloud for maintenance and computation,
can be either individual consumer or corporation.
2) CS (Cloud Server): an entity, which is managed by cloud service provider, has significant storage space and
computation resource to maintain the clients' data. 3) Combiner: an entity, which receives the storage request and distributes the block-tag pairs to the corresponding
cloud servers. When receiving the challenge, it splits the challenge and distributes them to the different cloud servers. When receiving the responses from the cloud servers, it combines them and sends the combined response to the verifier.
4) PKG (Private Key Generator): an entity, when receiving the identity, it outputs the corresponding private key.

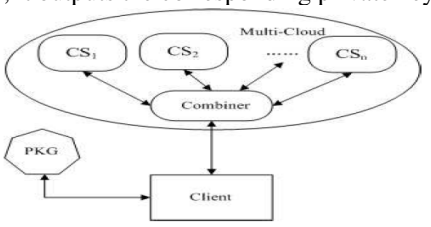


Fig 2 The System Model of ID-DPDP combiner

First, we give the definition of interactive proof system. It will be used in the definition of ID-DPDP. Then, we present the definition and security model of ID-DPDP protocol.
Definition 1 (Interactive Proof System): [22] Let c, s : N → R be functions satisfying c(n) > s(n) + 1 p(n) for some polynomial p(·). An interactive pair (P, V) is called a interactive proof system for the language L, with completeness bound c(·) and soundness bound s(·), if
1) Completeness: for every x ∈ L, Pr[< P, V > (x) =1] ≥ c(|x|).
2) Soundness: for every x 6∈ L and every interactive machine B, Pr[< B, V > (x) = 1] ≤ s(|x|).
Interactive proof system is used in the definition of IDDPDP, i.e., Definition 2.
Definition 2 (ID-DPDP): An ID-DPDP protocol is a collection of three algorithms (Setup, Extract, TagGen) and an interactive proof system (Proof). They are described in detail below.
1) Setup(1k): Input the security parameter k, it outputs the system public parameters params, the master public key mpk and the master secret key msk.
2) Extract(1k, params,mpk,msk, ID): Input the public parameters params, the master public key mpk, the master secret key msk, and the identity ID of a client, it outputs the private key skID that corresponds to the client with the identity ID.
3) TagGen(skID, Fi,P): Input the private key skID, the block Fi and a set of CS P = {CSj}, it outputs the tuple {φi, (Fi, Ti)}, where φi denotes the i-th record of metadata, (Fi, Ti) denotes the i-th block-tag pair. Denote all the metadata {φi} as φ.
Owner Registration: In this module an owner has to upload its files in a cloud server, he/she should register first. Then only he/she can be able to do it. For that he needs to fill the details in the registration form. These details are maintained in a database. Owner Login: In this module,any of the above mentioned person have to login,they should login by giving their emailid and password . User Registration: In this module if a user wants to access the data which is stored in a cloud, he/she should register their details first. These details are maintained in a Database. User Login: If the user is an authorized user, he/she can download the file by using file id which has been stored by data owner when it was uploading.

Access Control: Owner can permit access or deny access for accessing the data. So users can able to access his/her account by the corresponding data owner. If owner does not allow, user can't able to get the data.

III. ENCRYPTION & DECRYPTION:

Here we are using this aes_encrypt & aes_decrypt for encryption and decryption. The file we have uploaded which has to be in encrypted form and decrypt it

File Upload: In this module Owner uploads the file(along with meta data) into database, with the help of this metadata and its contents, the end user has to download the file. The uploaded file was in encrypted form, only registered user can decrypt it. File Download: The Authorized users can download the file from clou database. Cloud Service Provider Registration: In this module , if a cloud service provider(maintainer of cloud) wants to do some cloud offer , they should register first. Cloud Service Provider Login: After Cloud provider gets logged in, He/ She can see Cloud provider can view the files uploaded by their clients. Also upload this file into separate Cloud Database TTP (TRUSTED THIRD PARTY) LOGIN: In this module TTP has monitors the data owners file by verifying the data owner's file and stored the file in a database .Also ttp checks the CSP(CLOUD SERVICE PROVIDER),and find out whether the csp is authorized one or not.

Algorithm1: Computing users' machingimpact ranking

Input: The user-matching Data k check In.
Output: Impact Check Out r for all users.
1:for i=1 to n do
2: r0(i)=1/n
3: end for
4: δ =
5: $\in == \in$
6: while $\lambda > \in$
7: for I = 1 to n do

$$r_{k+1}(i) = \sum_j \frac{1-\varphi}{n} rk(j) + \varphi \sum_j \frac{w(i,j)\,rk}{\sum_j w(i,}$$

8:
9: end for

$$\lambda = \sum_1^n rk + 1(i) - rk$$

10:
11: end while

IV. THE PRIVACY PRESERVINGPUBLIC CLOUD SCHEME:

To effectively support public cloud without having to retrieve the data blocks themselves, we resort to the homomorphic authenticator technique [7,9,11]. Homomorphic authenticators are unforgeable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an auditor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. However, the direct adoption of these techniques is not suitable for our purposes, since the linear combination of blocks may potentially reveal user data information, thus violating the privacy-preserving guarantee. Specifically, if enough number of the linear combinations of the same blocks are collected, the TTP can simply derive the user's data content by solving a system of linear equations.

To achieve privacy-preserving public auditing, we propose to uniquely integrate the homomorphic authenticator with random mask technique. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by a pseudo random function (PRF). With random mask, the TTP no longer has all the necessary information to build up a correct group of linear equations and therefore cannot derive the user's data content, no matter how many linear combinations of the same set of file blocks can be collected. Meanwhile, due to the algebraic property of the homomorphic authenticator, the correctness validation of the block-authenticator pairs will not be affected by the randomness generated from a PRF, which will be shown shortly. Note that in our design, we use public key based homomorphic authenticator, specifically, the one in [11] which is based on BLS signature [16], to equip the auditing protocol with public auditability. Its flexibility in signature aggregation will further benefit us for the multi- task auditing.

Scheme Details Let G1, G2 and GT be multiplicative cyclic groups of prime order p, and e :G1 × G2 → GT be a bilinear map as introduced in preliminaries. Let g be the generator of G2.H(·) is a secure map-to-point hash function: {0, 1}* → G1, which maps strings uniformly to G1.

Another hash function h(·) : GT → Zp maps group element of GT uniformly to Zp.

Algorithm2: Anonymization algorithm

Input: T1, T2 a k-privacy requorement, a taxonomy tree for each categorical attribute in xn.
Output: a generalized T2 satiisfying the privacy require ment.

1. Generalize entry value of Ai to ANYwhere Ai€Xi
2. While there is a valid candidate in ᵁcut, do
3. Find the paire of near root (xi)from Úcut.
4. Specialized or on t2 and remove Xifrom Úcut.
5. Replace new (xi) and the valid status of xi for all in Úcut.
6. Out put the generalized T2 and Úcut.

V. SETUP PHASE:

1) The cloud user runs KeyGen to generate the system's public and secret parameters. He chooses a random x ← Zp, a random element u ← G1, and computes v ← gx. The secret parameter is sk = (x) and the public parameters are pk

= (v, g, u, e(u, v)). Given data file F = (m1, . . . ,mn),the user runs SigGen to compute signature _i for each block mi: _i ← (H(i) · umi)x ∈ G1 (i = 1, . . . , n). Denote the set of signatures by ∅ = sends {F, } to the server and deletes them from its local storage.

Distribution Phase:

2) During the verification process, to generate the audit message "chal", the TTP picks a random

c-element subset I = {s1, . . . , sc} of set [1, n], where sq = _kprp (q) for 1 ≤ q ≤ c and kprp is

the randomly chosen permutation key by TPA for each auditing. We assume that s1 ≤ · · · ≤ sc. For each element i ∈I, the TPA also chooses a random value _i (of a relative small bit length

compared to |p|). The message "chal" specifies the positions of the blocks that are required to be

checked in this Audit phase. The TPA sends the chal = {(i, _i)}i∈I to the server.

3) Upon receiving challenge chal = {(i, _i)}i∈I , the server runs GenProof to generate a response

proof of data storage correctness. Specifically, the server chooses a random element r ← Zp via

r = prf (chal), where kprf is the randomly chosen PRF key by server for each auditing, and

calculates R = e(u, v)r. Let μ′ denote the linear combination of sampled blocks specified in chal:

μ′ = ∑i∈I Vi, mi. To blind μ′ with r, the server computes: μ = r + μ′ mod p, where = h(R).

Meanwhile, the server also calculates an aggregated signature = Qi∈I (u,v)I i ∈ G1. It then sends {μ, v,R} as the response proof of storage correctness to the TPA. With the response from the server, the TPA runs VerifyProof to validate the response by first computing = h(R) and then checking the verification equation

R · e(, g) ?= e((sc

Y

i=s1

H(i) i) · u(μ, v) (1)

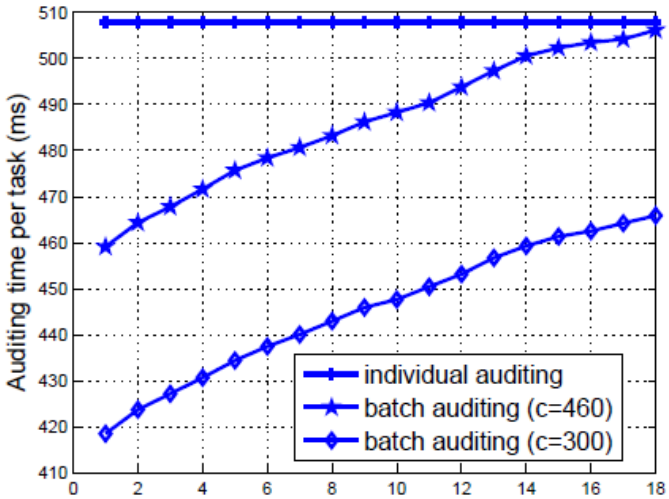
The correctness of the above verification equation can be elaborated as follows:

R · e(, g) = e(u, v)r . e((

Y

i=s1 (H(i) · umi)x· i), g) = e(ur, v) · e((

It is clear that the random mask r and related R = e(u, v)r has no effect on the validity of the checking result. The security of this protocol will be proved. Discussion As analyzed at the beginning of this section, this approach ensures the privacy of user data content during the auditing process. Meanwhile, the homomorphic authenticator helps achieve the constant communication overhead for server's response during the audit: the size of { , μ,R} is fixed and has nothing to do with the number of sampled blocks c. Note that there is no secret keying material or states for TPA to keep or maintain between audits, and the auditing protocol does not pose any potential on-line burden toward users. Since the TPA could "regenerate" the random c-element subset I = {s1, . . . , sc} of set [1, n], where sq = _kprp (q), for1 ≤ q ≤ c, unbounded usage is also achieved. Previous work [7,9] showed that if the server is missing a fraction of the data, then the number of blocks that needs to be checked in order to detect server misbehavior with high probability is in the order of O(1). For example, if the server is missing 1% of the data F, the TPA only needs to audit for c = 460 or 300 randomly chosen blocks of F so as to detect this misbehavior with probability larger than 99% or 95%, respectively. Given the huge volume of data outsourced in the cloud, checking a portion of the data file is more affordable and practical for both TPA and cloud server than checking all the data, as long as the sampling strategies provides high probability assurance. In Section 4, we will present the experiment result based on these sampling strategies.



Sorting out Invalid Responses Now we use experiment to justify the efficiency of our recursive binary search approach for TPA to sort out the invalid responses when batch auditing fails, as discussed in Section 3.4. Note that this experiment is tightly pertained to works by [17,24], which evaluates the batch verification efficiency of various short signature schemes. To evaluate the feasibility of the recursive approach, we first generate a collection of 256 valid responses, which implies the TPA may concurrently handle 256 different auditing delegations. We then conduct the tests repeatedly while randomly corrupting an - fraction, ranging from 0 to 18%, by replacing them with random values. The average auditing time per task against the individual auditing approach is presented in Fig. 3. The result shows that even the number of invalid responses exceeds 15% of the total batch size, the performance of batch auditing can still be safely concluded as more

preferable than the straightforward individual auditing. Note that is consistent with the experiment results derived in [17].

VI. RELATED WORK:

Ateniese et al. [7] are the first to consider public auditability in their defined "provable datapossession" (PDP) model for ensuring possession of data files on untrusted storages. Their scheme utilizes the RSA-based homomorphic authenticators for auditing outsourced data and suggests randomly sampling a few blocks of the file. However, the public auditability in their scheme demands the linear combination of sampled blocks exposed to external auditor.When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the auditor. Juels et al. [12] 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 a fixed priori, and public auditability is not supported in their main scheme. Although they describe a straightforwardMerkle-tree construction for public PoRs, this approach only works with encrypted data. Shacham et al. [11] design an improved PoR scheme built from BLS signatures with full proofs of security in the security model defined in [12]. Similar to the construction in [7], they use publicly verifiable homomorphic authenticators that are built from provably secure BLS signatures. Based on the elegant BLS construction, public retrievability is achieved. Again, their approach does not support privacy-preserving auditing for the same reason as [7]. Shah et al. [8,13] propose allowing a TPA to keep online storage honest by first encrypting the data then sending a number of pre-computed symmetric-keyed hashes over the encrypted data to the auditor. The auditor verifies both the integrity of the data file and the server's possession of a previously committed decryption key. This scheme only works for encrypted files, and it suffers from the auditor statefulness and bounded usage, which may potentially bring in on-line burden to users when the keyed hashes are used up. In other related work, Ateniese et al. [25] propose a partially dynamic version of the prior PDP scheme that uses only symmetric key cryptography. However, the system imposes a priori bound on the number of audits and does not support public auditability. In [18], Wang et al. consider a similar support for partial dynamic data storage in distributed scenario. The proposed challenge-response protocol can both determine the data correctness and locate possible errors.

In a subsequent work, Wang et al. [9] propose to combine BLS based homomorphic authenticator with MHT to support both public auditability and fully data dynamics. Almost simultaneously, Erway et al. [19] developed a skip lists based scheme to enable provable data possession with fully dynamics support. However, all their protocol requires the linear combination of sampled blocks just as [7, 11], and thus does not support privacy-preserving auditing on user's outsourced data. While all above schemes provide methods for efficient auditing and provable assurance on the correctness of remotely stored data, none of them meet all the requirements for privacy-preserving public auditing in Cloud Computing, as supported in our result. More importantly, none of these schemes consider batch auditing, which will greatly reduce the computation cost on the TPA when coping with large number of audit delegations.

VII. CONCLUSION:

In this work, we have identified a new privacy challenge during data accessing in the cloud computing to achieve privacy-preserving access authority sharing. Authentication is established to guarantee data confidentiality and data integrity. Data anonymity is achieved since the wrapped values are exchanged during transmission. User privacy is enhanced by anonymous access requests to privately inform the cloud server about the users' access desires. Forward security is realized by the session identifiers to prevent the session correlation. It indicates that the proposed scheme is possibly applied for enhanced privacy preservation in cloud applications.

References:

[1]. Mishra, R. Jain, and A. Durresi, "Cloud Computing: Networking and Communication Challenges," IEEE Communications Magazine, vol. 50, no. 9, pp, 24-25, 2012.

[2]. R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente,"Key Challenges in Cloud Computing to Enable the FutureInternet of Services," IEEE Internet Computing,[online] ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6203493, 2012.

[3]. K. Hwang and D. Li, "Trusted Cloud Computing with Secure Resources and Data Coloring," IEEE Internet Computing, vol. 14,no. 5, pp. 14-22, 2010.

[4]. J. Chen, Y. Wang, and X. Wang, "On-Demand Security Architecture for Cloud Computing," Computer, vol. 45, no. 7, pp. 73-78,2012.

[5]. Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multi-cloud Storage," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no, 12, pp.2231-2244, 2012.

[6]. H. Wang, "Proxy Provable Data Possession in Public Clouds,"IEEE Transactions on Services Computing, [online] ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6357181, 2012.

[7]. K. Yang and X. Jia, "An Efficient and Secure Dynamic AuditingProtocol for Data Storage in Cloud Computing,"

IEEE Transactions on Parallel and Distributed Systems, [online] ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6311398, 2012.

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

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

[10]. L. A. Dunning and R. Kresman, "Privacy Preserving Data Sharing With Anonymous ID Assignment," IEEE Transactions on Information Forensics and Security, vol. 8, no. 2, pp. 402-413, 2013.

[11]. X. Liu, Y. Zhang, B. Wang, and J. Yan, "Mona: Secure MultiOwner Data Sharing for Dynamic Groups in the Cloud," IEEE Transactions on Parallel and Distributed Systems, [online] ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6374615, 2012.

[12]. S. Grzonkowski and P. M. Corcoran, "Sharing Cloud Services:User Authentication for Social Enhancement of Home Networking,"IEEE Transactions on Consumer Electronics, vol. 57, no. 3, pp.1424-1432, 2011.

[13]. Y. Xiao, C. Lin, Y. Jiang, X. Chu, and F. Liu, "An Efficient Privacy-Preserving Publish-Subscribe Service Scheme for Cloud Computing," in Proceedings of Global Telecommunications Conference (GLOBECOM 2010), December 6-10, 2010.

[14]. H. Y. Lin and W. G. Tzeng, "A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 6, pp. 995-1003,2012.

[15]. J. Yu, P. Lu, G. Xue, and M. Li, "Towards Secure Multi-Keyword Top-k Retrieval over Encrypted Cloud Data," IEEETransactions on Dependable and Secure Computing, [online] ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6425381, 2013.

[16]. K. W. Park, J. Han, J. W. Chung, and K. H. Park, "THEMIS: AMutually Verifiable Billing System for the Cloud Computing Environment,"IEEE Transactions on Services Computing, [online] ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6133267, 2012.

[17]. R. Canetti, "Universally Composable Security: A New Paradigmfor Cryptographic Protocols," in Proceedings of the 42nd AnnualSymposium on Foundations of Computer Science (FOCS 2001), pp.136-145, October 14-17, 2001.