International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || REETA-2K16 ǁ PP. 773-788

A New Privacy-Preserving Public Auditing Using HLA Technique For Securing Cloud Data

M Amudha[1], R Priyadarshini[2]
[1]*Asst Professor DEPT OF CSE, SISTK-PUTTUR, AP*
[2]*Asst Professor DEPT OF CSE, SISTK-PUTTUR, AP*

ABSTRACT: Cloud storage, clients can remotely store their information a d appreciate the on-interest great applications and administrations from a related pool of configurable processing resources, without the weight of neighbourhood information storing and maintenance. Apart, the way that clients no more have physical ownership of the outsourced information makes the information honesty assurance in Distributed computing a considerable errand, particularly for clients with compelled registering assets. Besides, clients ought to have the capacity to quite recently utilize the distributed storage as though it is neighbourhood, without stressing over the need to check its respectability. Along these lines, empowering open auditability for distributed storage is of basic significance with the goal that clients can depend on an Third party auditor (TPA) to check the respectability of outsourced information and be straightforward. To safely present a compelling TPA, the inspecting procedure ought to get no new liabilities towards client information security, and inform no extra online weight with client. In this paper, we propose a protected cloud storage framework supporting privacy-preserving public auditing. We encourage extend our outcome to empower the TPA to perform reviews for various clients all the while and productively. Broad security and performance investigation demonstrate the proposed plans are provably secure and very proficient.
KEYWORDS: Data storage, privacy-preserving, public auditability, cryptographic protocols, TPA.

I. INTRODUCTION

Cloud computing has been envision because the next generation info technology (IT) design for associate enterprises, thanks to its lengthy list of distinctive benefits within the IT history: on-demand self-service, all over within the network access, location freelance resource pool, fast resource flexibility usage-based rating and transference of risk . As a disrupting technology with the profound implications, cloud computing may be a reworking the terribly nature of however business use info technology. One elementary facet of the paradigm dynamical the info square measure being centralized or outsourced to the cloud. From users' perspective, together with each people and IT enterprises, storing information remotely to the cloud in an exceedingly versatile on-demand approach brings engaging benefits: relief of the difficulty for storage management, universal information access with location independence, and shunning of the cost on hardware, software, and personnel maintenances, etc.While cloud computing makes the benefits a lot of appealing than ever, it additionally bring new and stringent security threats toward users' outsourced information. Since cloud service suppliers (CSP) square measure separate body entities, information outsourcing is really relinquishing user's final management over the fate of their information. As a result, the correctness of the info within the cloud is being place in danger thanks to the subsequent reasons. initial of all, though the infrastructures beneath the cloud square measure way more powerful and reliable than personal computing devices, they're still facing the broad vary of each internal and external threats for information integrity. samples of outages and security breaches of noteworthy cloud services seem from time to time. Second, there do exist numerous motivations for CSP to behave unreliably toward the cloud users concerning their outsourced information standing. as an examples, CSP would possibly reclaim storage for financial reasons by discarding information that haven't been or square measure seldom accessed, or maybe hide information loss incidents to keep up a name. In short, though outsourcing information to the cloud is economically engaging for long large-scale storage, it doesn't like a shot provide any guarantee on information integrity and convenience. This downside, if not properly self-addressed, might impede the success of cloud design.

II. PROBLEM STATEMENT

A. The Framework And Thread Model

We consider a cloud information stockpiling administration including three distinct substances, as showed in Fig. 1 the cloud client (U), who has huge measure of information documents to be put away in the cloud; the cloud server (CS), which is overseen by the cloud administration supplier (CSP) to give information stockpiling benefit and has critical storage room and calculation assets (we won't separate CS and CSP in the future); the outsider inspector (TPA), who has skill and capacities that cloud clients don't have and is trusted to

evaluate the distributed storage administration unwavering quality for the benefit of the client upon solicitation. Clients depend on the CS for cloud information stockpiling and upkeep. They may likewise progressively collaborate with the CS to get to and upgrade their put away information for different application purposes. To spare the computation asset and additionally the online weight, cloud clients may turn to TPA for guaranteeing the capacity trustworthiness of their outsourced information, while wanting to keep their information private from TPA.

We consider the presence of a semi-trusted CS as does. In particular, in the vast majority of time it carries on legitimately and does not go amiss from the endorsed convention execution. In any case, for their own advantages the CS may disregard to keep or purposely erase once in a while got to information documents which have a place with standard cloud clients. In addition, the CS may choose to conceal the information debasements brought about by server hacks or Byzantine disappointments to look after notoriety. We accept the TPA, who is in the matter of evaluating, is solid and autonomous, and therefore has no motivating force to plot with either the CS or the clients amid the examining procedure. Nonetheless, it hurts the client if the TPA could take in the outsourced information after the review. To approve the CS to react to the review delegated to TPA's, the client can sign an validation allowing review rights to the TPA's open key, and all reviews from the TPA are confirmed against such a certificate. These verification handshakes are precluded in the accompanying Presentation.

B.Design Goals

To enable privacy-preserving public auditing for cloud data storage under the aforementioned model, our protocol design should achieve the following security and performance guarantees.

1. Public auditability: to allow TPA to verify the correctness of the cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to the cloud users.

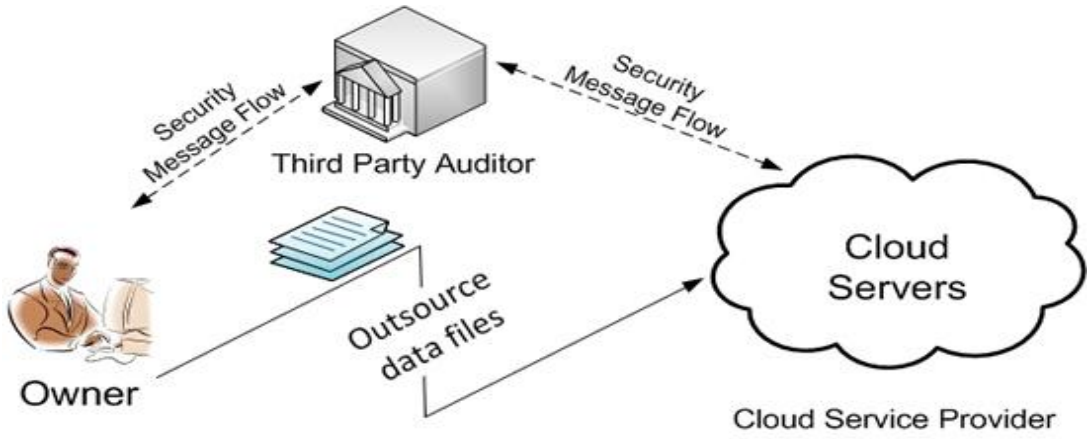

Fig. 1: The architecture of cloud data storage service

2) Storage correctness: to ensure that there exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.

3) Privacy-preserving: to ensure that the TPA can-not derive users' data content from the information collected during the auditing process.

4) Batch auditing: to enable TPA with secure and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different users simultaneously.

5) Lightweight: to allow TPA to perform auditing with minimum communication and computation overhead.

When we consider a cloud data storage service involving three different entities, the *cloud user* (U), who has large amount of data files to be stored in the cloud; the *cloudserver* (CS), which is managed by the *cloud service provider* (CSP) to provide data storage service and has significant storage space and computation resources (we will not differentiate CS and CSP hereafter); the [4] *third party auditor* (TPA), who has expertise

and capabilities that cloud users do Not have and is trusted to assess the cloud storage service reliability on behalf of the user upon request.

- As security threat is very high, which restrict user to use cloud computing.
- Existing mechanism for audit is not sufficient enough to handle audit.
- loss of control over data.
- Dependence on the Cloud Computing provider.

III THE PROPOSED SCHEMES

This section gives our public auditing scheme which affords a whole outsourcing solution of facts no longer simplest the statistics itself, however also its integrity take a look. We begin from an outline of our public auditing system and talk honest schemes and their demerits. Then we gift our essential scheme and display the way to volume our major scheme to assist batch auditing for the TPA upon delegations from multiple customers. Sooner or later, we speak how to generalize our privacy-keeping public auditing scheme and its guide of information dynamics.

1. Definitions and Framework

We follow a similar definition of previously proposed schemes in the context of remote data integrity checking and adapt the framework for our privacy preserving public auditing system. A public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof). KeyGen is a key generation algorithm that is run by the user to setup the scheme. SigGen is used by the user to generate verification metadata, which may consist of MAC, signatures, or other related information that will be used for auditing. GenProof is run by the cloud server to generate a proof of data storage correctness, while VerifyProof is run by the TPA to audit the proof from the cloud server.
Running a public auditing system consists of two phases, Setup and Audit:

• Setup: The user initializes the public and secret parameters of the system by executing KeyGen, and pre-processes the data file F by using SigGen to generate the verification metadata. The user then stores the data file F and the verification metadata at the cloud server, and delete its local copy. As part of pre-processing, the user may alter the data file F by expanding it or including additional metadata to be stored at server.

• Audit: The TPA issues an audit message or challenge to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will derive a response message from a function of the stored data file F and its verification metadata by executing GenProof. The TPA then verifies the response via VerifyProof.

Our framework assumes the TPA is stateless, which is a desirable property achieved by our proposed solution. It is easy to extend the framework above to capture a stateful auditing system, essentially by spliting the verification metadata into two parts which are stored by the TPA and the cloud server respectively.Our design does not assume any additional property on the data file. If the user wants to have more error-resiliency, he/she can always first redundantly encode the data file and then uses our system with the data file that has error-correcting codes integrated.

2. Notation and Preliminaries

This section presents our public auditing scheme which provides a complete outsourcing solution of data not only the data itself, but also its integrity checking. After introducing notations and brief preliminaries, we start from an overview of our public auditing system and discuss two straightforward schemes and their demerits. Then, we present our main scheme and show how to extent our main scheme to support batch auditing for the TPA upon delegations from multiple users. Finally, we discuss how to generalize our privacy-preserving public auditing scheme and its support of data dynamics.
F: the data file to be outsourced, denoted as a sequence of n blocks m1, m2, m3,..,mi,.., mn € Zp for some large prime p.
MAC(.) : message authentication code (MAC) function, defined as: K X {0, 1}* => {0, 1}l where K denotes Key space. H (.), h (.) : cryptographic hash functions
We now introduce some necessary cryptographic background for our proposed scheme.
Bilinear Map: Let G1, G2, and GT be multiplicative cyclic groups of prime order p. Let g1 and g2 be generators of G1 and G2, respectively. A bilinear map e is a map: G1 X G2 -> GT such that for all u € G1G2 and a; b €Zp, e(ua , vb) = e(u,v)ab.

This bilinearity implies that for any u1, u2 €G1, v€G2, e(u1.u2, v)= e(u1,v).e(u2; v). Of course, there exists an efficiently computable algorithm for computing e and the map should be nontrivial, i.e., e is non degenerate: e (g1; g2) ≠ 1.

3. The Basic Schemes

Before giving our main result, we study two classes of schemes as a warm-up. The first one is a MAC-based
solution which suffers from undesirable systematic demerits bounded usage and stateful verification, which may pose additional online burden to users, in a public auditing setting. This somehow also shows that the auditing problem is still not easy to solve even we have introduced a TPA. The second one is a system based on homomorphic linear authenticators (HLA), which covers many recent proof of storage systems. We will pinpoint the reason why all existing HLA-based systems are not privacy-preserving. The analysis of these basic schemes leads to our main result, which overcomes all these drawbacks. Our main scheme to be presented is based on a specific HLA scheme.

A) MAC-based Solution

There are two possible ways tomake use of MAC to authenticate the data. A trivial way is just uploading the data blocks with their MACs to the server, and sends the corresponding secret key sk to the TPA. Later, the TPA can randomly retrieve blocks with their MACs and check the correctness via sk. Apart from the high (linear in the sampled data size) communication and computation complexities, the TPA requires the knowledge of the data blocks for verification.

To circumvent the requirement of the data in TPA verification, one may restrict the verification to just consist of equality checking. The idea is as follows: Before data outsourcing, the cloud user chooses‘s random Message Authentication Code keys {skr}1<=r<=s’ precomputes MACs for the whole file F and publishes these verification metadata (the keys and the MACs) to TPA. The TPA can reveal a secret key skr to the cloud server and ask for a fresh keyed MAC for comparison in each audit. This is privacy preserving as long as it is impossible to recover Fin full given MAC skr(F) and skr. However, it suffers from the following severe drawbacks:

1) the number of times a particular data file can beaudited is limited by the number of secret keys that must be fixed a prior. Once all possible secret keys are exhausted, the user then has to retrieve data in full to recompute and republish new MACs to TPA.

2) The TPA also has to maintain and update statebetween audits, i.e., keep track on the revealed MAC keys. Considering the potentially large number of audit delegations from multiple users, maintaining such states for TPA can be difficultand error prone.

3) it can only support static data, and cannot efficiently deal with dynamic data at all. However, supporting data dynamics is also of critical importance for cloud storage systems.

B) HLA-based Solution

To effectively support public auditability without having to retrieve the data blocks themselves, the HLA technique can be used. HLAs, like MACs, are also some unforgeable verification metadata that authenticate the integrity of a data block. The difference is that HLAs can be aggregated. It is possible to compute an aggregated HLA which authenticates a linear combination of the individual data blocks. At a high level, an HLA-based proof of storage system works as follow. The user still authenticates each element of F = {mi} by a set of HLAs □. The TPA verifies the cloud storage by sending a random set of challenge {vi}. The cloud server then returns μ=Σivi. mi and its aggregated authenticator σ computed from □. Though allowing efficient data auditing and consuming only constant bandwidth, the direct adoption of these HLA based techniques is still not suitable for our purposes. This is because the linear combination of blocks, μ=Σivi. mi, may potentially reveal user data information to TPA, and violates the privacy-preserving guarantee. Specifically, by challenging the same set of c block m1, m2. . ., mc using c different sets of random coefficients {vi}, TPA can accumulate c different linear ombinations μ1, ... μc. With { μi} and {vi}, TPA can derive the user’s data m1,m2, . . ., mc by simply solving a system of linear equations.

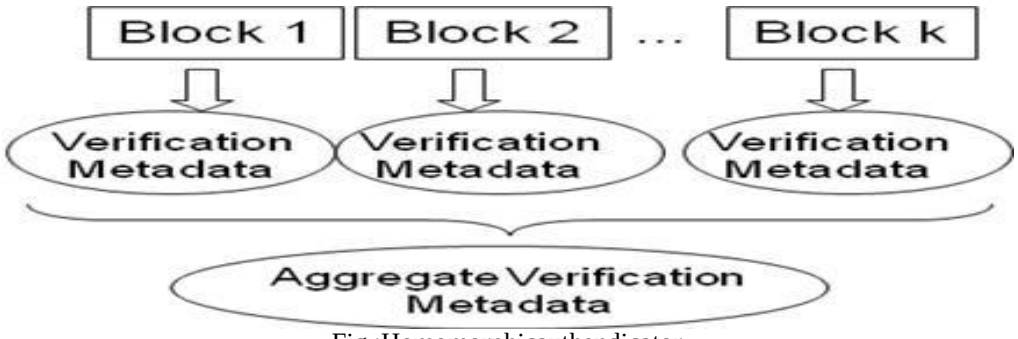

Fig :Homomorphicauthendicator

Block It splits as blocks n based on file size Verification Metadata – Verify the meta data of the file In addition to Aggregate Authenticator, a linear combination of file blocks re received by TPA

$$\mu = \sum_{i \in I} V_i m_i$$

vi are random number mi are file blocks TPA might be able to infer the file blocks, if it has many linear combinations of the same block Pseudo Random Function (PRF) provide a random mask that we can use.

4. Privacy-Preserving Public Auditing Scheme

Overview: To achieve privacy-preserving public auditing, we propose to uniquely integrate the homomorphic linear authenticator with random masking technique. In our protocol, the linear combination of sampled blocks in the server 's response is masked with randomness generated the server. With random masking, the TPA 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. On the other hand, the correctness validation of the block-authenticator pairs can still be carried out in a new way which will be shown shortly, even with the presence of the randomness. Our design makes use of a public key based HLA, to equip the auditing protocol with public auditability. Specifically, we use the HLA proposed in which is based on the short signature scheme proposed by Boneh, Lynn and Shacham .

Scheme details: Let G1, G2, and GT be multiplicativecyclic groups of prime order p, and e: G1 X G2 => GT be a bilinear map as introduced in preliminaries. Let g be agenerator of G2. H(.) is a secure map-to-point hash function: {0, 1}*=>G1, which maps strings uniformly toG1. Another hash function h(.) : GT =>Zp maps group element of GT uniformly to Zp. Our scheme is as follows:

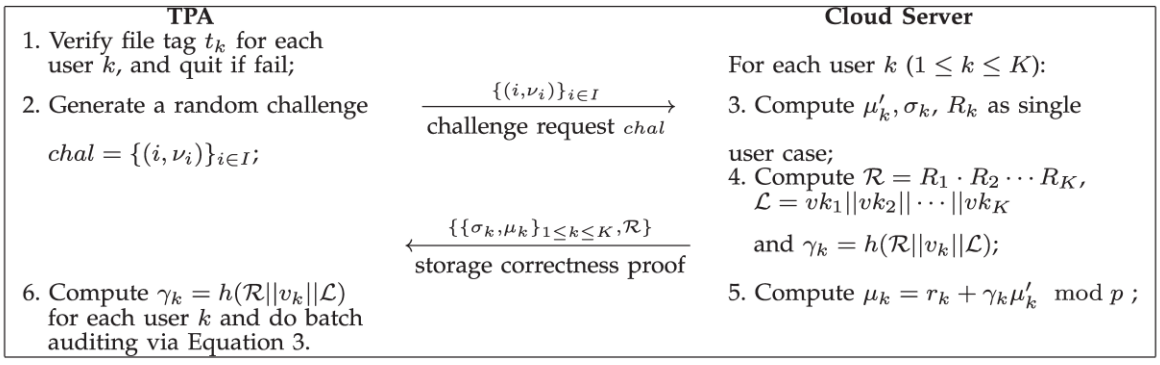

Fig: The privacy-preserving public auditing protocol

Setup Phase: The cloud user runs KeyGen to generate the public and secret parameters. Specifically, the user chooses a random signing key pair (spk, ssk), a random x <=Zp, a random element u <= G1, and computes v <= gx. The secret parameter is sk = (x, ssk) and the public parameters are pk = (spk, v, g, u; e(u, v)). Given a data file F= {mi}, the user runs SigGen to compute authenticator σ i <= H((Wi) . umi)x € G1 for each i. Here,

Wi = name || i and name is chosen by the user uniformly at random from Zp as the identifier of file F. Denote the set of authenticators by □ = { σ i}1<=i<=n. The last part of SigGen is for ensuring the integrity of the unique file identifier name. One simple way to do this is to compute t=name || SSigssk(name) as the file tag for F, where SSigssk(name) is the signature on name under the private key ssk. For simplicity, we assume the TPA knows the number of blocks n. The user then sends F along with the verification metadata (□, t) to the server and deletes them from local storage.

Audit Phase: The TPA first retrieves the file tag t. With respect to the mechanism we describe in the Setup phase, the TPA verifies the signature SSigssk(name) via spk, and quits by emitting FALSE if the verification fails. Otherwise, the TPA recovers name. Now it comes to the "core" part of the auditing process. To generate the challenge message for the audit "chal," the TPA picks a random c- element subset I = {s1, s2, s3,...,sc} of set [1, n]. For each element i € I, the TPA also chooses a random value vi. The message "chal" specifies the positions of the blocks required to be checked. The TPA sends chal = {(i, vi)} i€I to the server. Upon receiving challenge chal = {(i, vi)} i€I , the server runs GenProof to generate a response proof of data storage correctness. Specifically, the server chooses a random element r <= Zp, and calculates R = e(u, v)r € GT . Let μ' denote the linear combination of sampled blocks specified in chal: μ' = Σ i€Ivimi. To blind μ' with r, the server computes: μ = r +□ μ' mod p, where □= h(R) € Zp. Meanwhile, the server also calculates an aggregated authenticator σ = □ i€Iσivi € G1. It then sends {μ, σ, R}, as the proof of storage correctness to the TPA. With the response, the TPA runs VerifyProof to validate it by first computing □ = h(R) and then checking the verification equations.

5. Support for Batch Auditing

With the establishment of privacy-preserving public auditing, the TPA may concurrently handle multipleauditing upon different users' delegation. The individual auditing of these tasks for the TPA can be tedious and very inefficient. Given K auditing delegations on K distinct data files from K different users, it is more advantageous for the TPA to batch these multiple tasks together and audit at one time. Keeping this natural demand in mind, we slightly modify the protocol in a single user case, and achieves the aggregation of K verification equations (for K auditing tasks) into a single one, as shown in Equation . As a result, a secure batch auditing protocol for simultaneous auditing of multiple tasks is obtained.

IV RELATED WORK

Atenieseet *al.*are the first to consider public auditability in their defined "provable data possession" (PDP) model for ensuring possession of data files on untrusted storages. Their scheme utilizes the RSA based homomorphic linear 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. Juelset *al.*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 remotearchive 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. Although they describe a straightforward Merkle-tree construction for public PoRs, this approach only works with encrypted data. Dodisetal. give a study on different variants of PoR with private auditability. Shachamet *al.* design an improved PoR scheme built from BLS signatures with full proofs of security in the security model defined in Similar to the construction in they use publicly verifiable homomorphic linear authenticators that are built from provably secure BLS signatures. Based on the elegant BLS construction, a compact and public verifiable scheme is obtained. Again, their approach does not support privacy-preserving auditing for the same reason as]. Shah *et al.*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 online burden to users when the keyed hashes are used up. In other related work, Atenieseet *al.* propose a partially dynamic version of the prior PDP scheme, using only symmetric key cryptography but with a bounded number of audits. In Wanget *al.* consider a similar support for partial dynamic data . storage in a distributed scenario with additional feature of data error localization. In a subsequent work, Wang *et al.* propose to combine BLS-based HLA with MHT to support both public auditability and full data dynamics. Almost simultaneously, Erwayet *al.* developed a skip lists based scheme to enable provable data possession with full dynamics support. However, the verification in these two protocols requires the linear combination of sampled blocks just as and thus does not support privacy preserving auditing. While all the 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. More importantly, none of these schemes consider batch auditing, which can greatly reduce the computation cost on the TPA when coping with a large number of audit delegations.

V CONCLUSION

In this paper, we propose a privacy-preserving public auditing system for data storage security in cloud computing. We utilize the homomorphic linear authenticator and random masking to guarantee that the TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files, we further extend our privacy-preserving public auditing protocol into a multiuser setting, where the TPA can perform multiple auditing tasks in a batch manner for better efficiency. Extensive analysis shows that our schemes are provably secure and highly efficient. Our preliminary experiment conducted on Amazon EC2 instance further demonstrates the fast performance of our design on both the cloud and the auditor side. We leave the full-fledged implementation of the mechanism on commercial public cloud as an important future extension, which is expected to robustly cope with very large scale data and thus encourage users to adopt cloud storage services more confidently.

VI FUTURE ENHANCEMENTS

In our future work, we will investigate how to use entity level different key and that key is mailed through their registered mail address so that is useful them to see their data in the cloud server. Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files, we further extend our privacy-preserving public auditing protocol into a multiuser setting, where the TPA can perform multiple auditing tasks in a batch manner for better efficiency. Extensive analysis shows that our schemes are provably secure and highly efficient. Our preliminary experiment conducted on Amazon EC2 instance furtherdemonstrates the fast performance of our design on both the cloud and the auditor side. We leave the full fledged implementation of the mechanism on commercial public cloud as an important future extension, which is expected to robustly cope with very large scale data and thus encourage users to adopt cloud storage services more confidently.

VII REFERENCES

[1]. P. Mell and T. Grance, "Draft NIST working definition of cloud computing," Referenced on June. 3rd, 2009 Online ahttp://csrc.nist.gov/groups/SNS/cloud-computing/index. html, 2009.

[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, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. UCB-EECS-2009-28, Feb 2009.

[3]. M. Arrington, "Gmail disaster: Reports of mass email deletions," Online at http://www.techcrunch.com/2006/ 12/28/gmail-disasterreports-of-mass-email-deletions/, December 2006.

[4]. J. Kincaid, "MediaMax/TheLinkup Closes Its Doors," Online at http://www.techcrunch.com/2008/07/10/mediamaxthelinkup-closes-its-doors/, July 2008.

[5]. Amazon.com, "Amazon s3 availability event: July 20, 2008," Online at http://status.aws.amazon.com/s3 20080720.html, 2008.

[6]. S. Wilson, "Appengine outage," Online at http://www. cio-weblog.com/50226711/appengineoutage.php, June 2008.

[7]. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of CCS'07*, Alexandria, VA, October 2007, pp. 598–609.

[8]. M. A. Shah, R. Swaminathan, and M. Baker, "Privacypreserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[9]. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS'09, volume 5789 of LNCS*. Springer-Verlag, Sep. 2009, pp. 355–370.

[10]. A. Juels and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. of CCS'07*, Alexandria, VA, October 2007, pp. 584–597.

[11]. Cloud Security Alliance, "Security guidance for critical areas of focus in cloud computing," 2009, http://www. cloudsecurityalliance.org.

[12]. H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. of Asiacrypt 2008*, vol. 5350, Dec 2008, pp. 90–107.

[13]. M. A. Shah, M. Baker, J. C. Mogul, and R.Swaminathan, "Auditing to keep online storage services honest," in *Proc. OfHotOS'07*. Berkeley, CA, USA: USENIX Association, 2007, pp.

[14]. 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPPA)," Online at http://aspe.hhs.gov/admnsimp/pl104191.htm, 1996.

[15]. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access control in cloud computing," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, March 2010.