International Journal of Science Engineering and Advance Technology, IJSEAT, Vol 2, Issue 8, August - 2014 | ISSN 2321-6905

International Journal of Science Engineering and Advance Technology

Secrecy Stabilizing for Cloud Storage With Assessing Of Third Party

K Venkata Ramana#1, Murram Sree Harsha#2
#1 Department Of CSE,GOKULA KRISHNA COLLEGE OF ENGINEERING & TECHNOLOGY,SULLURPET.
#2Student Of M.Tech(C.S) And Department Of CSE,GOKULA KRISHNA COLLEGE OF ENGINEERING & TECHNOLOGY,SULLURPET

Abstract

Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Cloud Computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public auditability for cloud storage is of critical importance so that users can resort to a third party auditor (TPA) to check the integrity of outsourced data and be worry-free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy, and introduce no additional online burden to user. In this paper, we propose a secure cloud storage system supporting privacy-preserving public auditing. We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient.

Index Terms—Data storage, privacy-preserving, public auditability, cryptographic protocols, cloud computing.

1 INTRODUCTION

CLOUD Computing has been envisioned as the next-generation information technology (IT) architecture for enterprises, due to its long list of unprecedented advantages in the IT history: on-demand self-service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage-based pricing and transference of risk [1]. As a disruptive technology with profound implications, Cloud Computing is transforming the very nature of how businesses use information technology. One fundamental aspect of this paradigm shifting is that data is being centralized or outsourced to the Cloud. From users' perspective, including both individuals and IT enterprises, storing data remotely to 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].

While Cloud Computing makes these advantages more appealing than ever, it also brings new and challenging security threats towards users' outsourced data. Since cloud service providers (CSP) are separateadministrative entities, data outsourcing is actually 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]–[7]. Secondly, there do exist various motivations for CSP to behave unfaithfully towards the cloud users regarding the status of their outsourced data. For examples, CSP might reclaim storage for monetary reasons by discarding data that has not been or is rarely accessed, or even hide data loss incidents so as to maintain a reputation [8]–[10]. In short, although outsourcing data to the cloud is economically attractive for long-term large-scale data storage, it does not immediately offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the successful deployment of the cloud architecture.

To fully ensure the data integrity and save the cloud users' computation resources as well as online burden, it is of critical importance to enable public auditing service for cloud data storage, so that users may resort to an independent third party auditor (TPA) to audit the outsourced data when needed. The TPA, who has expertise and capabilities that users do not, can periodically check the integrity of all the data stored in the cloud on behalf of the users, which provides a much more easier and affordable way for the users to ensure their storage correctness in the cloud. Moreover, in addition to help users to evaluate the risk of their subscribed cloud data services, the audit result from TPA would also be beneficial for the cloud service providers to improve their cloud based service platform, and even serve for independent arbitration purposes [9]. In a word, enabling public auditing services will play an important role for this nascent cloud economy to become fully established, where users will need ways to assess risk and gain trust in the cloud.

Specifically, our contribution 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 enables an external auditor to audit user's outsourced data in the cloud without learning 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. Specifically, 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.

The rest of the paper is organized as follows. Section II introduces the system and threat model, and our design goals. Then we provide the detailed description of our scheme in Section III. Section IV gives the security analysis and performance evaluation, followed by Section V which overviews the related work. Finally, Section VI gives the concluding remark of the whole paper.

2 PROBLEM STATEMENT

2.1 The System and Threat Model

We consider a cloud data storage service involving three different entities, as illustrated in Fig. 1: the *cloud user* (U), who has large amount of data files to be stored in the cloud; the *cloud server* (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 *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. Users rely on the CS for cloud data storage and maintenance. They may also dynamically interact with the CS to access and update their stored data for various application purposes. To save the computation resource as well as the online burden, cloud users may resort to TPA for ensuring the storage integrity of their outsourced data, while hoping to keep their data private from TPA.

We consider the existence of a semi-trusted CS as [16] does. Namely, in most of time it behaves properly and does not deviate from the prescribed protocol execution. However, for their own benefits the CS might neglect to keep or deliberately delete rarely accessed data files which belong to ordinary cloud users. Moreover, the CS may decide to hide the data corruptions caused by server hacks or Byzantine failures to maintain reputation. We assume the TPA, who is in the business of auditing, is reliable and independent, and thus has no incentive to collude with either the CS or the users during the auditing process. However, it harms the user if the TPA could learn the outsourced data after the audit.

To authorize the CS to respond to the audit delegated to TPA's, the user can sign a certificate granting audit rights to the TPA's public key, and all audits from the TPA are authenticated against such a certificate. These authentication handshakes are omitted in the following presentation.

2.2 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.

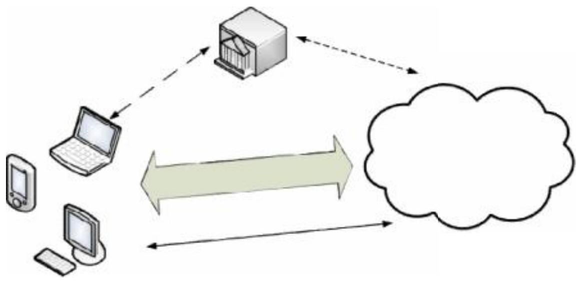

Fig. 1: The architecture of cloud data storage service

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 cannot 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.

3 T HE PROPOSED SCHEMES

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. 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.

3.1 Definitions and Framework

We follow a similar definition of previously proposed schemes in the context of remote data integrity checking [8], [11], [13] 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 deletes 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 encodes the data file and then uses our system with the data file that has error-correcting codes integrated.

3.2 Support for Data Dynamics

In Cloud Computing, outsourced data might not only be accessed but also updated frequently by users for various application purposes [10], [19] – [21]. Hence, supporting data dynamics for privacypreserving public auditing is also of paramount importance. Now we show how to build upon the existing work [10] and adapt our main scheme to support data dynamics, including block level operations of modification, deletion and insertion.

3.3 Generalization

As mentioned before, our protocol is based on the HLA in [13]. Recently, it has been shown in [23] that HLA can be constructed by homomorphic identification protocols. One may apply the random masking technique we used to construct the corresponding zero knowledge proof for different homomorphic identification protocols. Therefore, it follows that our privacy-preserving public auditing system for secure cloud storage can be generalized based on other complexity assumptions, such as factoring [23].

4 EVALUATION

4.1 Security Analysis

We evaluate the security of the proposed scheme by analyzing its fulfillment of the security guarantee described in Section 2.2, namely, the storage correctness and privacy-preserving property. We start from the single user case, where our main result is originated. Then we show the security guarantee of batch auditing for the TPA in multi-user setting.

4.2 Performance Analysis

We now assess the performance of the proposed privacy-preserving public auditing schemes to show that they are indeed lightweight. We will focus on the cost of the efficiency of the privacy-preserving protocol and our proposed batch auditing technique. The experiment is conducted using C on a Linux system with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM, and a 7200 RPM Western Digital 250 GB Serial ATA drive with an 8 MB buffer. Our code uses the Pairing-Based Cryptography (PBC) library version 0.4.18. The elliptic curve utilized in the experiment is a MNT curve, with base field size of 159 bits and the embedding degree 6. The security level is chosen to be 80 bit, which means $|v_i| = 80$ and $|p| = 160$. All experimental results represent the mean of 20 trials.

4.2.1 Cost of Privacy-Preserving Protocol

We begin by estimating the cost in terms of basic cryptographic operations, as notated in Table 1. Suppose there are c random blocks specified in the challenge

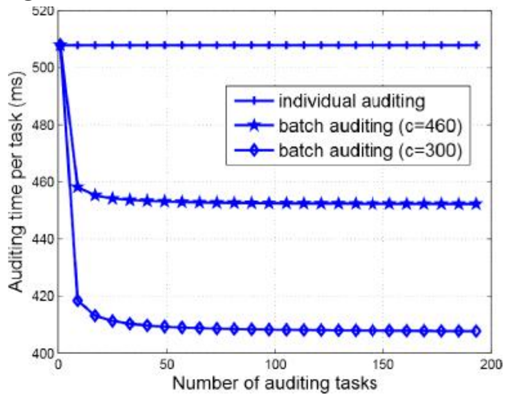

Fig. 2: Comparison on auditing time between batch and individual auditing. Per task auditing time denotes the total auditing time divided by the number of tasks. For clarity reasons, we omit the straight curve for individual auditing when c=300.

message *chal* during the Audit phase. Under this setting, we quantify the cost introduced of the privacypreserving auditing in terms of server computation, auditor computation as well as communication overhead.

On the server side, the generated response includes an aggregated authenticator $\sigma = \prod_{i \in I} \sigma_i^{\nu_i} \in \mathbb{G}_1$, r a random factor $R = e(u,v) \in G_T$, and a blinded linear combination of sampled blocks $\mu = \gamma \ P_{i \in I}\, v_i m_i + r \in Z_p$, where $\gamma = h(R) \in Z_p$. The corresponding computation cost is c-$MultExp^1_{G1}(|v_i|)$, $Exp^1_{GT}(|p|)$, and $Hash^1_{\mathbb{Z}_p} + Add^c_{\mathbb{Z}_p} + Mult^{c+1}_{\mathbb{Z}_p}$, respectively. Compared to the existing HLA-based solution for ensuring remote data integrity [13][1], the extra cost for protecting the user privacy, resulted from the random mask R, is only a constant: $Exp^1_{\mathbb{G}_T}(|p|) + Mult^1_{\mathbb{Z}_p} + Hash^1_{\mathbb{Z}_p} +$ Add^1_{Zp}, which has nothing to do with the number of sampled blocks c. When c is set to be 300 to 460 for high assurance of auditing, as discussed in Section 3.4, the extra cost for privacy-preserving guarantee on the server side would be negligible against the total server computation for response generation.

Similarly, on the auditor side, upon receiving the response $\{\sigma,R,\mu\}$, the corresponding computation cost for response validation is $Hash^1_{Z_p} + c\text{-}MultExp^1_{G_1}(|v_i|) + Hash^c_{G_1} + Mult^1_{G_1} + Mult^1_{G_T} + Exp^3_{G_1}(|p|) + Pair^2_{G_1,G_2}$, among which only $Hash^1_{Z_p} + Exp^2_{G_1}(|p|) + Mult^1_{G_T}$ account for the additional constant computation cost. For c = 460 or 300, and considering the relatively expensive pairing operations, this extra cost imposes little overhead on the overall cost of response validation, and thus can be ignored. For the sake of completeness, Table 2 gives the experiment result on performance comparison between our scheme and the state-of-the-art [13]. It can be

1. We refer readers to [13] for a detailed description of their HLAbased solution.

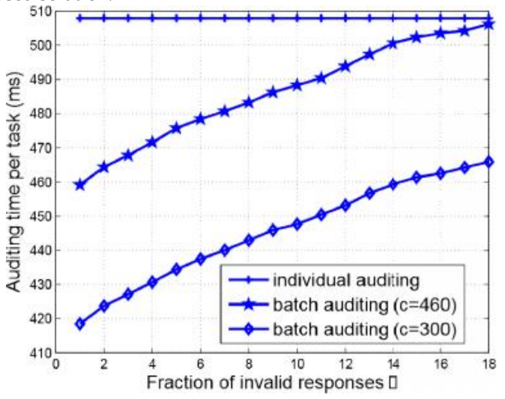

Fig. 3: Comparison on auditing time between batch and individual auditing, when α-fraction of 256 responses are invalid. Per task auditing time denotes the total auditing time divided by the number of tasks.

shown that the performance of our scheme is almost the same as that of [13], even if our scheme supports privacy-preserving guarantee while [13] does not. For the extra communication cost of our scheme opposing to [13], the server's response $\{\sigma,R,\mu\}$ contains an additional random element R, which is a group element of G_T and has the size close to 960 bits.

4.2.2 Batch Auditing Efficiency

Discussion in Section 3.5 gives an asymptotic efficiency analysis on the batch auditing, by considering only the total number of pairing operations. However, on the practical side, there are additional less expensive operations required for batching, such as modular exponentiations and multiplications. Meanwhile, the different sampling strategies, i.e., different number of sampled blocks c, is also a variable factor that affects the batching efficiency. Thus, whether the benefits of removing pairings significantly outweighs these additional operations is remained to be verified. To get a complete view of batching efficiency, we conduct a timed batch auditing test, where the number of auditing tasks is increased from 1 to approximately 200 with intervals of 8. The performance of the corresponding non-batched (individual) auditing is provided as a baseline for the measurement. Following the same experimental settings c = 300 and c = 460, the average per task auditing time, which is computed by dividing total auditing time by the number of tasks, is given in Fig. 4 for both batch and individual auditing. It can be shown that compared to individual auditing, batch auditing indeed helps reducing the TPA's computation cost, as more than 11% and 14 % of per-task auditing time is saved, when c is set to be 460 and 300, respectively.

4.2.3 Sorting out Invalid Responses

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 αfraction, 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. 5. 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 the random distribution of invalid responses within the collection is nearly the worst-case for batch auditing. If invalid responses are grouped together, it is possible to achieve even better results.

5 RELATED WORK

Ateniese *et al.* [8] 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 RSAbased 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. Juels *et al.* [11] 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. Although they describe a straightforward Merkle-tree construction for public PoRs, this approach only works with encrypted data. Dodis *et al.* [25] give a study on different variants of PoR with private auditability. Shacham *et al.* [13] design an improved PoR scheme built from BLS signatures [17] with full proofs of security in the security model defined in [11]. Similar to the construction in [8], 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 [8]. Shah *et al.* [9], [14] 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, Ateniese *et al.* [19] propose a partially dynamic version of the prior PDP scheme, using only symmetric key cryptography but with a bounded number of audits. In [20], Wang *et 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.* [10] propose to combine BLS-based HLA with MHT to support both public auditability and full data dynamics. Almost simultaneously, Erway *et al.* [21] 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 [8], [13], and thus does not support privacypreserving 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 privacypreserving 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.

6 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 multi-user 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.

REFERENCES

[1] P. Mell and T. Grance, "Draft NIST working definition of cloud computing," Referenced on June. 3rd, 2009 Online at http://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/appengine outage.php, June 2008.

[7] B. Krebs, “Payment Processor Breach May Be Largest Ever,” Online at http://voices.washingtonpost.com/securityfix/ 2009/01/payment _ processor breach may b.html, Jan. 2009.

[8] 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.

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

[10] 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.

[11] 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.

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

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

[14] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, “Auditing to keep online storage services honest,” in *Proc. of HotOS’07*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1 – 6.

[15] 104th United States Congress, “Health Insurance Portability and Accountability Act of 1996 (HIPPA),” Online at http:// aspe.hhs.gov/admnsimp/pl104191.htm, 1996.

[16] 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.

[17] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the Weil pairing,” *J. Cryptology*, vol. 17, no. 4, pp. 297–319 , 2004.

[18] A. L. Ferrara, M. Greeny, S. Hohenberger, and M. Pedersen, “Practical short signature batch verification,” in *Proceedings of CT-RSA, volume 5473 of LNCS*. Springer-Verlag, 2009, pp. 309 – 324.

[19] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, “Scalable and efficient provable data possession,” in *Proc. of SecureComm’08*, 2008, pp. 1–10.

[20] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring data storage security in cloud computing,” in *Proc. of IWQoS’09*, July 2009, pp. 1 – 9.

[21] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, “Dynamic provable data possession,” in *Proc. of CCS’09*, 2009 , pp. 213–222.

[22] R. C. Merkle, “Protocols for public key cryptosystems,” in *Proc. of IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA, 1980.

[23] G. Ateniese, S. Kamara, and J. Katz, “Proofs of storage from homomorphic identification protocols,” in *ASIACRYPT*, 2009 , pp. 319–333.

[24] M. Bellare and G. Neven, “Multi-signatures in the plain publickey model and a general forking lemma,” in *ACM Conference on Computer and Communications Security*, 2006, pp. 390–399.

[25] Y. Dodis, S. P. Vadhan, and D. Wichs, “Proofs of retrievability via hardness amplification,” in *TCC*, 2009, pp. 109–127.

K.Venkata Ramana,
Associate Professor,
Department Of Cse,
Gokula Krishna College Of Engineering & Technology
Sullurpet.

M.Sreeharsha,
M. Tech(Cse)
Student,
Department Of Cse,
Gokula Krishna College Of Engineering & Technology
Sullurpet.