www.ijsetr.com

National Conference on
CASH LESS TRANSACTIONS USING COMPUTER AND COMMUNICATIONS
www.worldconferences.org

VIGNANA BHARATHI Institute of Technology
(Approved by AICTE, Accredited by NBA & NAAC, Permanently Affiliated to JNTUH)

Integrity Auditing and Data Deduplication on Cloud using Secured Systems

DAVARAPALLI ANANDAM
Assistant Professor, Dept of CSE, PACE Institute of Tech and Sciences, Vallur, Ongole, AP, India,
E-mail: anandbabu.1361@gmail.com.

Abstract: The cloud computing technology develops during the last few years, outsourcing data to cloud service for storage, which benefits in sparing efforts on heavy data maintenance and management. The outsourced cloud storage is not fully trustworthy, it raises security concerns on how to realize data deduplication in cloud while obtaining integrity auditing. In this work, we study the problem of integrity auditing and secure deduplication on cloud data. Specifically, aiming at obtaining both data integrity and deduplication in cloud, we propose two secure systems, namely SecCloud and SecCloud+. SecCloud introduces an auditing entity which helps clients to generate data tags before uploading as well as audit the integrity of data having been stored in cloud. Compared with previous work, the computation by user in SecCloud is greatly reduced during the file uploading and auditing phases. SecCloud+ is designed for that customers always want to encrypt their data before uploading, and enables integrity auditing and secure deduplication on encrypted data.

Keywords: SecCloud, Data Integrity, Deduplication, SecCloud+.

I. INTRODUCTION

Cloud storage is used for data storagewhere data is stored in virtualized pools of storage which are generally organized by third parties. Cloud storage provides customers with benefits, ranging from cost savings and simplified convenience. These great features attract more and more customers to use and storage their personal data to the cloud storage. According to the latest analysis report, the volume of data in cloud is expected to achieve 40 trillion gigabytes in 2020.Even though cloud storage system has been widely used, it fails to accommodate some important emerging needs such as the abilities of auditing integrity of cloud files by cloud clients and detecting duplicated files by cloud servers. We illustrate both problems below.The first problem is integrity auditing. The cloud serveris able to reduce clients from the heavy burden of storage management and maintenance. The most difference of cloud storage from traditional datacenter storage is that the data is transferred via Internet and stored in an uncertain domain, not under control of the clients at all, which unavoidably raises clients great concerns on the integrity of their data. These concerns originate from the fact that the cloud storage issusceptible to security threats from both outside and inside of the cloud [1], and the uncontrolled cloud from the clients to maintain their reputation. What is more serious is that for saving money and space, the cloud servers might even actively and deliberately discard rarely accessed data files belonging to an ordinary client. Considering the large size of the outsourced data files and the clients' constrained resource capabilities, the first problem is generalized as how can the client efficiently perform periodical integrity verifications even without the local copy of data files.

The second problem is secure deduplication. The fast adoption of cloud services is accompanied by increasing volumes of data stored at remote cloud servers. Among these remote stored files, most of them are duplicated: according to a recent survey by EMC [2], 75% of recent digital data is duplicated copies. This fact raises a technology namely deduplication, in which the cloud servers would like to deduplicate by keeping only a single copy for each file (or block) and make a link to the file (or block) for every client who owns or asks to store the same file (or block). Unfortunately, this action of deduplication would lead to a number of threatspotentially affecting the storage system [3] [2]. In this paper, achieving data integrity and deduplication in cloud, we propose two secure systems namely SecCloud and SecCloud+. SecCloud introduces an auditing entity with a maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud. This design fixes the issue of previous work that the computational load at user or auditor is too huge for tag generation. For completeness of fine-grained, the functionality of auditing designed in SecCoud is supported on both block level and sector level. In addition, SecCoud also enables secure deduplication. That the security considered in SecCoud is the prevention of leakage of side channel information. In order to prevent the leakage of such side channel information, we follow the tradition of [3][2] and design a proof of ownership protocol between clients and cloud servers, which allows clients to prove to cloud servers that they exactly own the target data.

Motivated by the fact that customers always want to encrypt their data before uploading, for reasons ranging from personal privacy to corporate policy, we introduce a key server into SecCloud as with [4] and propose the SecCloud+ schema. Besides supporting integrity auditing and secure deduplication, SecCloud+ enables the guarantee of file confidentiality. Specifically, thanks to the property of deterministic encryption in convergent encryption, we propose a method of directly auditing integrity on encrypted data. The challenge of deduplication on encrypted is the prevention of dictionary attack [4]. We make a modification on convergent encryption such that the convergent key of file is generated and controlled by a secret seed, such that any adversary could not directly derive the convergent key from the content of file and the dictionary attack is prevented. This paper is organized as follows: In Section II, we review the related works on integrity auditing and secure deduplication. Section III and Section IV respectively proposes the SecCloud and SecCoud+ system. Finally Section V draws the conclusion of this paper.

II.RELATED WORK

My work is related to both integrity auditing and secure deduplication, we review the works in both areas in the following subsections, respectively.

A. Integrity Auditing

The definition of provable data possession (PDP) was introduced by Ateniese et al. [5][6] for assuring that the cloud servers possess the target files without retrieving or downloading the whole data. Essentially, PDP is a probabilistic proof protocol by sampling a random set of blocks and asking the servers to prove that they exactly possess these blocks, and the verifier only maintaining a small amount of metadata is able to perform the integrity checking. After Ateniese et al.'s proposal [5], several works concerned on how to realize PDP on dynamic scenario: Ateniese et al. [7] proposed a dynamicPDP schema but without insertion operation; Erway et al. [8] improved Ateniese et al.'s work [7] and supported insertion by introducing authenticated flip table; A similar work has also been contributed in [9]. Nevertheless, these proposals [5][7][8][9] suffer from the computational overhead for tag generation at the client. To fix this issue, Wang et al. [10] proposed proxy PDP in public clouds. Zhu et al. [11] proposed the cooperative PDP in multi-cloud storage. Another line of work supporting integrity auditing is proof of retrievability (POR) [12]. Compared with PDP, POR not merely assures the cloud servers possess the target files, but also guarantees their full recovery. In [12], clients apply erasure codes and generate authenticators for each block for verifiability and retrievability. In order to achieve efficient data dynamics, Wang et al. [13] improved the POR model by manipulating the classic Merkle hash tree construction for block tag authentication. Xu and Chang [14] proposed to improve the POR schema in [12] with polynomial commitment for reducing communication cost. Stefanov et al. [15] proposed POR protocol over authenticated file system subject to frequent changes. Azraoui et al. [16] combined the privacy-preserving word search algorithm with the insertion in data segments of randomly generated short bit sequences, and developed a new POR protocol. Li et al. [17] considered a new cloud storage architecture with two independent cloud servers for integrity auditing to reduce the computation load at client side. Recently, Li et al. [18] utilized the key-disperse paradigm to fix the issue of a significant number of convergent keys in convergent encryption.

B. Secure Deduplication

Deduplication is a technique where the server stores only a single copy of each file, regardless of how many clients asked to store that file, such that the disk space of cloud servers as well as network bandwidth are saved. However, trivial client side deduplication leads to the leakage of side channel information. For example, a server telling a client that it need not send the file reveals that some other client has the exact same file, which could be sensitive information in some case. In order to restrict the leakage of side channel information, Halevi et al. [3] introduced the proof of ownership protocol which lets a client efficiently prove to a server that that the client exactly holds this file. Several proof of ownership protocols based on the Merkle hash tree are proposed [3] to enable secure client-side deduplication. Pietro and Sorniotti[19] Proposed an efficient proof of ownership scheme by choosing the projection of a file onto some randomly selected bit-positions as the file proof. nother line of work for secure deduplication focuses on the confidentiality of deduplicated data and considers to make deduplication on encrypted data. Ng et al. [20] firstly introduced the private data deduplication as a complement of public data deduplication protocols of Halevi et al. [3].

Convergent encryption [21] is a promising cryptographic primitive for ensuring data privacy in deduplication. Bellare et al. [22] formalized this primitive as message-locked encryption, andexplored its application in space-efficient secure outsourced storage. Abadi et al. [23] further strengthened Bellare et al's security definitions [22] by considering plaintext distributions that may depend on the public parameters of the schemas.Regarding the practical implementation of convergent encryption for securing deduplication, Keelveedhi et al. [4] designed the DupLESS system in which clients encrypt under file-based keys derived from a key server via an oblivious pseudorandom function protocol. As stated before, all the works illustrated above considers either integrity auditing or deduplication, while in this paper, we attempt to solve both problems simultaneously. In addition, it is worthwhile noting that our work is also distinguished with [2] which audits cloud data with deduplication, because we also consider to 1) outsource the computation of tag generation, 2) audit and deduplicate encrypted data in the proposed protocols.

III. SECCLOUD

In this section, I describe my proposed SecCloud system. Specifically, I begin with giving the system model of

SecCloud as well as introducing the design goals for SecCloud. In what follows, I detail.

A. System Model

Aiming at allowing for auditable and deduplicated storage,we propose the SecCloud system. In the SecCloud system, we have three entities:

- Cloud Clients have large data files to be stored andrely on the cloud for data maintenance and Computation.
- Cloud Servers virtualises the resources according to the requirements of clients and expose them as storage pools. Typically, the cloud clients may buy or rent storage capacity from cloud servers, and store their individual data in these bought or rented spaces for future utilization.
- Auditor which helps clients upload and audit their outsourced data maintains a MapReduce cloud and acts like a certificate authority. This assumption presumes that the auditor is associated with a pair of public and private keys. Its public key is available to the other entities in the system. The SecCloud system supporting file-level deduplication includes the following three protocols respectively highlighted by red, blue and green in Fig. 1

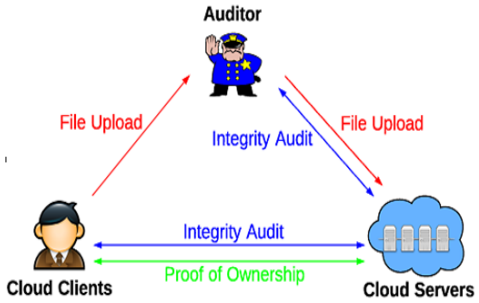


Fig 1: SecCloud Architecture.

B. File Uploading Protocol: This protocol is used to allowing clients to upload files via the auditor. Specifically, the file uploading protocol includes three phases:

- **Phase 1** (cloud client → cloud server): client performs the duplicate check with the cloud server to confirm if such a file is stored in cloud storage or not before uploading a file. If there is a duplicate, another protocol called Proof of Ownership will be run between the clientand the cloud storage server. Otherwise, the following protocols (including phase 2 and phase 3) are run between these two entities.
- **Phase 2** (cloud client → auditor): client uploads files to the auditor, and receives a receipt from auditor.
- **Phase3**(auditor → cloud server): auditor helps generate a set of tags for the uploading file, and send them along with this file to cloud server.

C. Integrity Auditing Protocol: It is an interactive protocol for integrity verification and allowed to be initialized by any entity except the cloud server. In this protocol, the cloud server plays the role of prover, while the auditor or client works asthe verifier. This protocol includes two phases:

- **Phase 1** (cloud client/auditor → cloud server): verifier (i.e., client or auditor) generates a set of challenges and sends them to the prover (i.e., cloud server).
- **Phase 2** (cloud server → cloud client/auditor): based on the stored files and file tags, prover (i.e., cloud server) tries to prove that it exactly owns the target file by sending the proof back to verifier (i.e., cloud client or auditor).At the end of this protocol, verifier outputs true if the integrity verification is passed.

D. Proof of Ownership Protocol: It is an interactive protocol initialized at the cloud server for verifying that the client exactly owns a claimed file. This protocol is typically triggered along with file uploading protocol to prevent the leakage ofside channel information. On the contrast to integrity auditing protocol, in PoW the cloud server works as verifier, while the client plays the role of prover. This protocol also includes two phases.

- **Phase 1** (cloud server → client): cloud server generates a set of challenges and sends them to the client.
- **Phase 2** (client → cloud server): the client responds with the proof for file ownership, and cloud server finally verifies the validity of proof.

Our Main Objectives are Outlined as Follows:

- **Integrity Auditing:** The first design goal of this work is to provide the capability of verifying correctness of the remotely stored data. The integrity verification further requires two features:

1. Public verification, which allows anyone, not just the clients originally stored the file, to perform verification;
2.Stateless verification, which is able to eliminate the need for state information maintenance at the verifier side between the actions of auditing and data storage

- **Secured Deduplication:** The second design goal of this work is secured deduplication. In other words, it requires that the cloud server is able to reduce the storage space by keeping only one copy of the same file. Notice that, regarding to secure deduplication, our objective is distinguished from previous work [3] in that we propose a method for allowing both deduplication over files and tags.
- **Cost-Effective:** The computational overhead for providing integrity auditing and secure deduplication should not represent a major additional cost to traditional cloud storage, nor should they alter the way either uploading or downloading operation.

IV. SECCLOUD+

I specify that my proposed SecCloud system has achieved both integrity auditing and file deduplication. However, it cannot prevent the cloud servers from knowing the content of files having been stored. In other words, the functionalities of integrity auditing and secure deduplication are only imposed on plain files. In this section, we propose SecCloud+, which allows for integrity auditing and deduplication on encrypted files.

A. System Model

Compared with SecCloud, our proposed SecCloud+ involves an additional trusted entity, namely key server, which is responsible for assigning clients with secret key (according to the file content) for encrypting files. This architecture is in line with the recent work [4]. But our work is distinguished with the previous work [4] by allowing for integrity auditing on encrypted data. SecCloud+ follows the same three protocols (i.e., the file uploading protocol, the integrity auditing protocol and the proof of ownership protocol) as with SecCloud. The only difference is the file uploading protocol in SecCloud+ involves an additional phase for communication between cloud client and key server. That is, the client needs to communicate with the key server to get the convergent key for encrypting the uploading file before the phase 2 in SecCloud. Unlike SecCloud, another design goals of file confidentiality is desired in SecCloud+ as follows.

- **File Confidentiality:** The design goal of file confidentiality requires to prevent the cloud servers from accessing the content of files. Specially, we require that the goal of file confidentiality needs to be resistant to "dictionary attack". That is, even the adversaries have pre-knowledge of the "dictionary" which includes all the possible files, they still cannot recover the target file [4].

B. SecCloud+ Details

We introduce the system setup phase of SecCloud+ as Follows:

- **System Setup:** As with SecCloud, the auditor initializes the public key pk = (gα, {ui}ti=1) and private key sk = α, where g, u1, u2, . . . , ut∈R G. In addition, to preserve the confidentiality of files, initially, the key server picks a random key ks for further generating file encryption keys, and each client is assigned with a secret key ck for encapsulating file encryption keys. Based on the initialized parameters, we then respectively describe the three protocols involved in SecCloud+.
- **File Uploading Protocol:** Suppose the uploading file F has s blocks, say B1, B2, . . . , Bs, and each block Bi for i = 1,2, . . . , s contains t sectors, say Bi1, Bi2, . . . , Bit.Client computes hF = Hash(F) by itself. In addition,for each sector Bij of F where i = 1,2, . . . , s and j =1,2, . . . , t, client computes its hash hBij = Hash(Bij). Finally(hF, {hBij}i=1,...,s,j=1,...,t) is sent to key server for generatingthe convergent keys for F. Upon receiving the hashes, the key server computes sskF =f(ks, hF) and sskij = f(ks, hBij) for i = 1, . . . , s andj = 1, . . . , t, where ks is the convergent key seed kept at the key server, and f(.) is a pseudorandom function. It is worthwhile noting that,
  - We take advantage of the idea of convergent encryption [21][22][23] to make the deterministic and "content identified" encryption, in which each "content" (file or sector) is encrypted using the session key derived from itself. In this way, different "contents" would result in different ciphertexts, and deduplication works.
  - Convergent encryption suffers from dictionary attack, which allows the adversary to recover the whole content with a number of guesses. To prevent such attack, as with [4], a "seed" (i.e., convergent key seed) is used for controlling and generating all the convergent keys to avoid the fact that adversary could guess or derive the convergent key just from the content itself.
  - We generate convergent keys on sector-level (i.e.,generate convergent keys for each sector in file F), to enable integrity auditing. Specifically, since convergent encryption is deterministic, it allows to compute homomorphic signatures on (convergent) encrypted data as with on plain data, and thus the sector-level integrity auditing is preserved. Client then continues to encrypt F sector by sector and uploads the ciphertext to auditor. Specifically, for each sector Bij of F, i = 1,2, . . . , s and j = 1,2, . . . , t, client computes ctBij=Enc(sskBij,Bij), and sends (IDF, {ctBij} i=1,...,s,j=1,...,t) to auditor, where Enc(.) is the symmetric encryption algorithm. The convergent keys sskij are encapsulated by client's secret key ck and directly stored at the cloud servers.

C. Proof of Ownership Protocol: Suppose a client claims that he/she has a file F and wants to store it at the cloud server, where F is an existing file having been stored on the server. The client needs to show the proof that he owns the same file at local. The user performs the proof of ownership in a similar way as [3] based on the encrypted file. If it is passed, a pointer will be provided to the client for the accessto the same file stored in the cloud server.

D. Security Definitions: Based on the paradigm of SecCloud and SecCloud+, we define the security definitions, adapting to the integrity auditing and secure deduplication goals. Our both definitions capture the philosophy of game-based definition. Specifically, we define two games respectively for integrity auditing and secure deduplication, and both of the games are played by two players, namely adversary and challenger. The adversary (the role of which is worked by semi-honest cloud server and cloud client respectively in integrity auditing and secure deduplication definition) is trying to achieve the goal condition explicitly specified in the game. Having this intuition, we give our security definitions as follows.

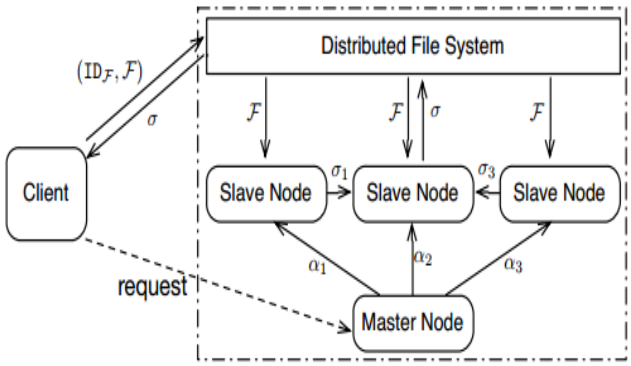
1. Integrity Auditing: An integrity auditing protocol is sound if any cheating cloud server that convinces the verifier that it is storing a file F is actually storing this file. To capture this spirit, we define its game based on Proof of Retrievability (PoR). The security model called Proof of Retrievability (PoR) was introduced by Shacham and Waters' in [12]. This security model captures the requirement for integrity auditing, whose basic security goal is to achieve proof of retrievability. In more details, in this security model, if there exists an adversary who can forge and generate any valid integrity proofs for any file F with a non-negligible probability, another simulator can be constructed who is able to extract F

with overwhelming probability. The formal definition for the above model can be given by the following game between a challenger and an adversary A. Note that in the following security game, the challenger plays the role of auditing server while the adversary A acts as the storage server.

- **Setup Phase:** The challenger runs the setup algorithm with required security parameter and other public parameter as input. Then, it generates the public and secret key pair (pk, sk). The public key pk is forwarded to the adversary.
- **Query phase:** The adversary is allowed to query the file upload oracle for any file F. Then, the file with the correct tags are generated and uploaded to the cloud storage server. These tags can be publicly verified with respect to the public key pk.
- **Challenge Phase:** A can adaptively send file F to the file tag tag comes, C runs the integrity verification protocol IntegrityVerify{A C(pk, tag)} with A.
- **Forgery:** A outputs a file tag tag′ and the description of a prover Pt. We say that a proverPt on tag′ is β-admissible, if the following two conditions hold:
  1. tag′ is a file tag output by a previous upload query.
  2. Pr[IntegrityVerify{Pt C(pk, tag′)} = 1] ≥ β.

Then we can define the soundness of PoR

Definition 3: (Proof of Retrievability) A PoR scheme is (β, γ)-sound if for any β-admissible proverPt output by A in the above game, there exists an extractor E that can recover the original file of tag tag with probability at least 1 − γ.



1. Secure Deduplication: Similarly, we can also define a game between challenger and adversary for secure deduplication below. Notice that the game for secure deduplication captures the intuition of allowing the malicious client to claim it has a challenge file F through colluding with all the other clients not owning this file.

- **Setup Phase:** A challenge file F with fixed length and minimum entropy (specified in system parameter) is randomly picked and given to the challenger. The challenger continues to run a summary algorithm and generate a summary sumF.
- **Learning Phase:** Adversary F can setup arbitrarily many client accomplices not exactly having F and have them to interact with the cloud servers to try to prove the ownership of file F. Notice that in the learning phase, the cloud server plays as the honest verifier withinput sum sumF and the accomplices could follow any arbitrary protocol set by A.
- **Challenge Phase:** The exact proof of ownership protocol is executed. Specifically, the challenger outputs a challenge to A and A responses with a proof based on its learnt knowledge. If A's proof is accepted by the cloud server, we say A succeeds. The security in terms of secure deduplication is achieved, if for all probabilistic polynomial-time adversaries A, the probability that A succeeds in the above experiment is negligible.

V. CONCLUSION

Aim to achieving both data integrity and deduplication in cloud, I propose SecCloud and SecCloud+. SecCloud introduces an auditing entity with maintenance of a MapReduce cloud, which helps clients generate data tags before uploadingas well as audit the integrity of data having been stored in cloud. In addition, SecCoud enables secure deduplication through introducing a Proof of Ownership protocol and preventing the leakage of side channel information in data deduplication. Compared with previous work, the computation byuser in SecCloud is greatly reduced during the file uploading and auditing phases. SecCloud+ is an advanced construction motivated by the fact that customers always want to encrypt their data before uploading, and allows for integrity auditing and secure deduplication directly on encrypted data.

VI. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski,G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Communication of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[2] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in IEEE Conference on Communications and Network Security (CNS), 2013, pp. 145–153.

[3] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in Proceedings of the 18th ACM Conference on Computer and Communications Security. ACM, 2011, pp. 491–500.

[4] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Serveraided encryption for deduplicated storage," in Proceedings of the 22Nd USENIX Conference on Security, ser. SEC'13. Washington, D.C.: USENIX Association, 2013, pp.179–194.[Online].Available:https://www.usenix.org/conference/usenixsecurity13/technicalsessions/presentation/bellare

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proceedings of the 14th ACM Conference on Computer and Communications Security, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 598– 609.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable datapossession," ACM Trans. Inf. Syst. Secur., vol. 14, no. 1, pp. 12:1–12:34, 2011.

[7] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, ser. SecureComm '08. New York, NY, USA: ACM, 2008, pp. 9:1–9:10.
[8] C. Erway, A. Kupc¸ ¨ u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proceedings of the 16th ACM Conference on Computer and Communications Security, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 213–222.
[9] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and ´J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Trans. on Knowl. and Data Eng., vol. 20, no. 8, pp. 1034–1038, 2008.
[10] H. Wang, "Proxy provable data possession in public clouds," IEEE Transactions on Services Computing, vol. 6, no. 4, pp. 551–559, 2013.
[11] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 12, pp. 2231– 2244, 2012.
[12]H. Shacham and B. Waters, "Compact proofs of retrievability," inProceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, ser. ASIACRYPT '08. Springer Berlin Heidelberg, 2008, pp.90–107.
[13] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Computer Security – ESORICS 2009, M. Backes and P. Ning, Eds., vol. 5789. Springer Berlin Heidelberg, 2009, pp. 355–370.
[14] J. Xu and E.-C. Chang, "Towards efficient proofs of retrievability," in Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ser. ASIACCS '12. New York, NY, USA: ACM, 2012, pp.79–80.
[15] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea, "Iris: A scalable cloud file system with efficient integrity checks," in Proceedings of the 28th Annual Computer Security Applications Conference, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 229–238.
[16] M. Azraoui, K. Elkhiyaoui, R. Molva, and M. Onen, "Stealthguard: Proofs of retrievability with hidden watchdogs," in Computer Security - ESORICS 2014, ser. Lecture Notes in Computer Science, M. Kutyłowski and J. Vaidya, Eds., vol. 8712. Springer International Publishing, 2014, pp. 239–256.
[17] J. Li, X. Tan, X. Chen, and D. Wong, "An efficient proof of retrievability with public auditing in cloud computing," in 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS), 2013, pp. 93–98.
[18] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 6, pp. 1615–1625, June 2014.
[19] R. Di Pietro and A. Sorniotti, "Boosting efficiency and security in proof of ownership for deduplication," in Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ser. ASIACCS '12. New York, NY, USA: ACM, 2012, pp. 81–82.
[20] W. K. Ng, Y. Wen, and H. Zhu, "Private data deduplication protocols in cloud storage," in Proceedings of the 27th Annual ACM Symposium on Applied Computing, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 441–446.
[21] J. Douceur, A. Adya, W. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in 22nd International Conference on Distributed Computing Systems, 2002, pp. 617–624.
[22] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in Advances in Cryptology – EUROCRYPT 2013, ser. Lecture Notes in Computer Science, T. Johansson and P. Nguyen, Eds. Springer Berlin Heidelberg, 2013, vol. 7881, pp. 296–312.
[23] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in Advances in Cryptology – CRYPTO 2013, ser. Lecture Notes in Computer Science, R. Canetti and J. Garay, Eds. Springer Berlin Heidelberg, 2013, vol. 8042, pp. 374–391.
[24] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in Advances in Cryptology — CRYPTO 2001, ser. Lecture Notes in Computer Science, J. Kilian, Ed. Springer Berlin Heidelberg, 2001, vol. 2139, pp. 213–229.

Author's Profile:

Davarapalli Anandam, Assistant Professor, PACE Institute of Technology and having 7 years teaching Experience.