ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

Dynamic Proof of Storage Deduplication for Multi-User Environments

M.Mrudhula, M.V Manikanth

Pursuing M.Tech from Prakasham Engineering College, Prakasam, Andhra Pradesh, India

M.Tech, Prakasham Engineering College, Prakasam, Andhra Pradesh, India

ABSTRACT: Dynamic Proof of Storage (PoS) is a useful cryptographic primitive that enables a user to check the integrity of outsourced files and to efficiently update the files in a cloud server. Although researchers have proposed many dynamic PoS schemes in single-user environments, the problem in multi-user environments has not been investigated sufficiently. A practical multi-user cloud storage system needs the secure client-side cross-user deduplication technique, which allows a user to skip the uploading process and obtain the ownership of the files immediately, when other owners of the same files have uploaded them to the cloud server. To the best of our knowledge, none of the existing dynamic PoSs can support this technique. In this paper, we introduce the concept of deduplicatable dynamic proof of storage and propose an efficient construction called DeyPoS, to achieve dynamic PoS and secure cross-user deduplication, simultaneously. Considering the challenges of structure diversity and private tag generation, we exploit a novel tool called Homomorphic Authenticated Tree (HAT). We prove the security of our construction, and the theoretical analysis and experimental results show that our construction is efficient in practice.

KEYWORDS: Cloud storage, dynamic proof of storage, deduplication.

I. INTRODUCTION

Storage outsourcing is becoming more and more attrac-tive to both industry and academia due to the advan-tages of low cost, high accessibility, and easy sharing. Asone of the storage outsourcing forms, cloud storage gains wide attention in recent years [1] [2]. Many companies, such as Amazon, Google, and Microsoft, provide their own cloud storage services, where users can upload their files to the servers, access them from various devices, and share them with the others. Although cloud storage services are widely adopted in current days, there still remain many security issues and potential threats [3] [4].

Data integrity is one of the most important properties when a user outsources its files to cloud storage. Users should be convinced that the files stored in the server are not tampered. Traditional techniques for protecting data integrity, such as message authentication codes (MACs) and digital signatures, require users to download all of the files from the cloud server for verification, which incurs a heavy communication cost [5]. These techniques are not suitable for cloud storage services where users may check the integrity frequently, such as every hour [6]. Thus, re-searchers introduced Proof of Storage (PoS) [7] for checking the integrity without downloading files from the cloud server. Furthermore, users may also require several dynamic operations, such as modification, insertion, and deletion, to update their files, while maintaining the capability of PoS.Dynamic PoS [8] is proposed for such dynamic operations. In contrast with PoS, dynamic PoS employs authenticated structures [9], such as the Merkle tree [10]. Thus, when dy- namic operations are executed, users regenerate tags (whichare used for integrity checking, such as MACs and signatures) for the updated blocks only, instead of regenerating for all blocks.

To better understand the following contents, we present more details about PoS and dynamic PoS. In these schemes [5] [8] [11], each block of a file is attached a(cryptographic) tag which is used for verifying the integrity of that block. When a verifier wants to check the integrity of a file, it randomly selects some block indexes of the file, and sends them to the cloud server. According to these chal-lenged indexes, the cloud server returns the corresponding blocks along with their tags. The verifier checks the block integrity and index correctness. The former can be directly guaranteed by

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

cryptographic tags. How to deal with the latter is the major difference between PoS and dynamic PoS.In most of the PoS schemes [5] [11] [12], the block index is "encoded" into its tag, which means the verifier can check the block integrity and index correctness simultaneously.However, dynamic PoS cannot encode the block indexes into tags, since the dynamic operations may change many indexes of non-updated blocks, which incurs unnecessary computation and communication cost. For example, there is a file consisting of 1000 blocks, and a new block is inserted behind the second block of the file. Then, 998 block indexes of the original file are changed, which means the user has to generate and send 999 tags for this update. Authenticated structures are introduced in dynamic PoSs [8] [13] [14] to solve this challenge. As a result, the tags are attached to the authenticated structure rather than the block indexes.Taking the Merkle tree in Fig. 1a as an example (Merkle tree is one of the most efficient authenticated structures in dynamic PoS [14]), the tag corresponding to the second file block involves the index of the Merkle tree node v5, that is 5, rather than 2. When a new block is inserted behind the second file block, the authenticated structure turns into the structure in Fig. 1b. Then, the index in the tag corresponding to the second file block changes, and the user only has to generate 2 tags for this update. This figure provides an instance that authenticated structure used in dynamic PoS reduces the computation cost in the update process.

However, dynamic PoS remains to be improved in a multi-user environment, due to the requirement of cross-user deduplication on the client-side [15]. This indicates that users can skip the uploading process and obtain the ownership of files immediately, as long as the uploaded files already exist in the cloud server. This technique can reduce storage space for the cloud server [16], and save transmission bandwidth for users. To the best of our knowledge, there is no dynamic PoS that can support secure cross-user deduplication.

There are two challenges in order to solve this problem.On one hand, the authenticated structures used in dynamic PoSs, such as skip list [8] and Merkle tree [14], are not suitable for deduplication. We call this challenge structure diversity, which means the authenticated structure of a file in dynamic PoS may have some conflicts. For instance, the au-thenticated structure of a file F is shown in Fig. 1a. When the file is updated to F, the authenticated structure stored on the server-side may turn into the structure in Fig. 1b. How-ever, an owner who intends to upload F usually generates a structure as shown in Fig. 1c, which is different from the structure stored in the cloud server. Thus, the owner cannot execute deduplication unless the owner and the cloud server synchronize the authenticated structure. On the other hand, even if cross-user deduplication is achieved (for example,the cloud server sends the entire authenticated structure to the owner), private tag generation is still a challenge for dynamic operations. In most of the existing dynamic PoSs,a tag used for integrity verification is generated by the secret key of the uploader. Thus, other owners who have the ownership of the file but have not uploaded it due to the cross-user deduplication on the client-side, cannot generate a new tag when they update the file. In this situation, the dynamic PoSs would fail.

If we take dynamic PoS and cross-user deduplication on the client-side as orthogonal issues, we may simply combine the existing dynamic PoS schemes and deduplication tech- niques. Then, structure diversity is solved via deduplication scheme. For solving private tag generation, each owner can generate its own authenticated structure and upload the structure to the cloud server, which means that the cloud server stores multiple authenticated structures for each file.Also, when a file is updated by a user, the cloud server has to update the corresponding authenticated structure in dynamic PoS, and construct a new authenticated structure for deduplication. As a result, this trivial combination in-troduces unnecessary computation and storage cost to thecloud server. Taking the combination of [10] and [15] as example, [10] is a dynamic PoS scheme which employs Merkle tree as its authenticated structure, and [15] is a cross-user deduplication scheme which also employs Merkle tree as its authenticated structure. Suppose Alice and Bob in-dependently own a file F, a Merkle tree TF is generated and stored by the cloud server for deduplication, and two Merkle trees TA and TB are generated by Alice and Bob respectively, and stored in the cloud server for PoS. When Alice updates F to F , the cloud server updates TA to T A for PoS and generates a new Merkle tree TF ′ for deduplication. Thus, the number of Merkle trees grows with the version numbers and the number of owners, which is 4 (TF , T A, TB, and TF ′) in the above example. Also, the cloud server has to generate two Merkle trees in the above example which is more time-consuming than update the Merkle trees. As a summary, existing dynamic PoSs cannot be extended to the multi-user environment.

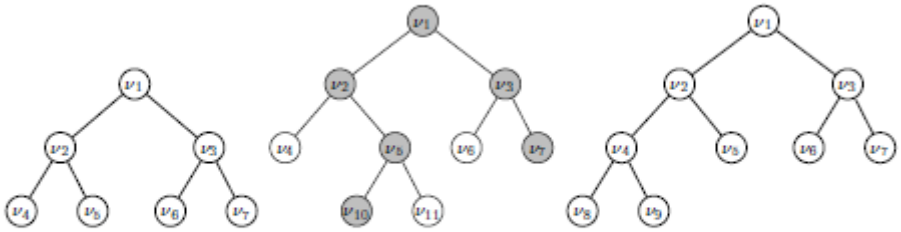
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017



(a) Initialize F (b) Update F to F' (c) Initialize F'

Fig. 1. An Overview of Tree-based Authenticated Structures

II. RELATED WORK

The concept of proof of storage was introduced by Ateniese et al. [5], and Juels and Kaliski [17], respectively. The main idea of PoS is to randomly choose a few data blocks as the challenge. Then, the cloud server returns the challenged data blocks and their tags as the response. Since the data blocks and the tags can be combined via homomorphic functions, the communication costs are reduced. The sub- sequent works [11] [18] [19] [20] [21] [22] [23] [24] [25] extended the research of PoS, but those works did not take dynamic operations into account. Erway et al. [8] and later works [13] [14] [26] [27] [28] [29] focused on the dynamic data. Among them, the scheme in [14] is the most efficient solution in practice. However, the scheme is stateful, which requires users to maintain some state information of their own files locally. Hence, it is not appropriate for a multi- user environment.

Halevi et al. [15] introduced the concept of proof of ownership which is a solution of cross-user deduplication on the client-side. It requires that the user can generate the Merkle tree without the help from the cloud server, which isa big challenge in dynamic PoS. Pietro and Sorniotti [30] proposed another proof of ownership scheme which im-proves the efficiency. Xu et al. [31] proposed a client-side deduplication scheme for encrypted data, but the scheme employs a deterministic proof algorithm which indicates that every file has a deterministic short proof. Thus, anyone who obtains this proof can pass the verification without possessing the file locally. Other deduplication schemes for encrypted data [32] [33] [34] were proposed for en-hancing the security and efficiency. Note that, all existing techniques for cross-user deduplication on the client-side were designed for static files. Once the files are updated, the cloud server has to regenerate the complete authenticated structures for these files, which causes heavy computation cost on the server-side.

Zheng and Xu [35] proposed a solution called proof of storage with deduplication, which is the first attempt to design a PoS scheme with deduplication. Du et al. [36] introduced proofs of ownership and retrievability, which are similar to [35] but more efficient in terms of computation cost. Note that neither [35] nor [36] can support dynamic operations. Due to the problem of structure diversity and private tag generation, [35] and [36] cannot be extended to dynamic PoS.

Wang et al. [37] [38], and Yuan and Yu [39] considered proof of storage for multi-user updates, but those schemes focus on the problem of sharing files in a group. Dedu- plication in these scenarios is to deduplicate files among different groups. Unfortunately, these schemes cannot sup- port deduplication due to structure diversity and private tag generation. In this paper, we consider a more general situation that every user has its own files separately. Hence,we focus on a deduplicatable dynamic PoS scheme in multi-user environments.

The major techniques used in PoS and dynamic PoS schemes are homomorphic Message Authentica-tion Codes [40] and homomorphic signatures [41] [42].With the help of homomorphism, the messages and MACs/signatures in these schemes can be compressed into a single message and a single MAC/signature. Therefore,the communication cost can

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

be dramatically reduced. These techniques have been used in PoS [7] [14] [18] and secure network coding [43] [44] [45]. A brief survey of homomor-phic MACs and signatures could be referred in [46].

1.2 Contributions

The main contributions of this paper are as follows.

1) To the best of our knowledge, this is the first work to introduce a primitive called deduplicatable dynamic Proof of Storage (deduplicatable dynamic PoS), which solves the structure diversity and private tag gener-ation challenges.

2) In contrast to the existing authenticated structures,such as skip list [8] and Merkle tree [14], we designa novel authenticated structure called Homomorphic Authenticated Tree (HAT), to reduce the communica-tion cost in both the proof of storage phase and the deduplication phase with similar computation cost.Note that HAT can support integrity verification, dynamic operations, and cross-user deduplication with good consistency.

3) We propose and implement the first efficient con-struction of deduplicatable dynamic PoS called Dey-PoS, which supports unlimited number of verifi-cation and update operations. The security of this construction is proved in the random oracle model,and the performance is analyzed theoretically and experimentally.

1.3 Organization

The rest of this paper is organized as follows. In Section 2,we introduce the models of deduplicatable dynamic proof of storage. An authenticated structure, called HAT, is designed in Section 3. In Section 4, we propose a concrete scheme, named DeyPoS. The security analysis and performance evaluation results are presented in Section 5 and Section 6, respectively. In the last section, we conclude this paper.

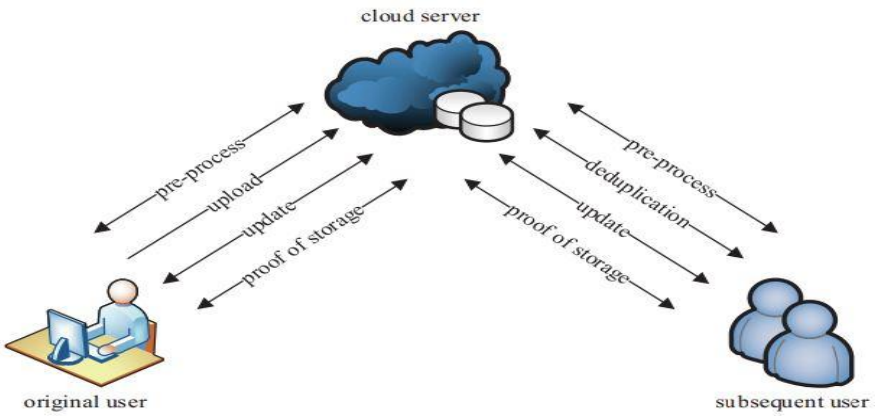


Fig.2 The system model of Deduplicable dynamic Pos

III. PROPOSED ALGORITHM

DEDUPLICATABLE DYNAMIC POS

As discussed in Section 1, no trivial extension of dynamic PoS can achieve cross-user deduplication. To fill this void, we present a novel primitive called deduplicatable dynamic proof of storage in this section.

2.1 System Model

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

Our system model considers two types of entities: the cloud server and users, as shown in Fig. 2. For each file, originaluser is the user who uploaded the file to the cloud server, while subsequent user is the user who proved the wnership of the file but did not actually upload the file to the cloud server. There are five phases in a deduplicatable dynamic PoS system: pre-process, upload, deduplication, update, and proof of storage. In the pre-process phase, users intend to upload their local files. The cloud server decides whether these files should be uploaded. If the upload process is granted, go into the upload phase; otherwise, go into the deduplication phase. In the upload phase, the files to be uploaded do not exist in the cloud server. The original users encodes the local files and upload them to the cloud server. In the deduplication phase, the files to be uploaded al-ready exist in the cloud server. The subsequent users possess the files locally and the cloud server stores the authenticated structures of the files. Subsequent users need to convince the cloud server that they own the files without uploading them to the cloud server. Note that, these three phases (pre-process, upload, and deduplication) are executed only once in the life cycle of a file from the perspective of users. That is, these three phases appear only when users intend to upload files. If these phases terminate normally, i.e., users finish uploading in the upload phase, or they pass the verification in the deduplication phase, we say that the users have the own- erships of the files.

In the update phase, users may modify, insert, or delete some blocks of the files. Then, they update the corre-sponding parts of the encoded files and the authenticated structures in the cloud server, even the original files were not uploaded by themselves. Note that, users can update the files only if they have the ownerships of the files,which means that the users should upload the files in the upload phase or pass the verification in the deduplication phase. For each update, the cloud server has to reserve the original file and the authenticated structure if there exist other owners, and record the updated part of the file and the authenticated structure. This enables users to update a file concurrently in our model, since each update is only "attached" to the original file and authenticated structure. In the proof of storage phase, users only possess a small constant size metadata locally and they want to check whether the files are faithfully stored in the cloud server without downloading them. The files may not be uploaded by these users, but they pass the deduplication phase and prove that they have the ownerships of the files. Note that, the update phase and the proof of storage phase can be executed multiple times in the life cycle of a file. Once the ownership is verified, the users can arbitrarily enter the update phase and the proof of storage phase without keeping the original files locally.

3 HOMOMORPHIC AUTHENTICATED TREE

Overview

To implement an efficient deduplicatable dynamic PoS scheme, we design a novel authenticated structure called homomorphic authenticated tree (HAT). A HAT is a binary tree in which each leaf node corresponds to a data block. Though HAT does not have any limitation on the number of data blocks, for the sake of description simplicity, we assume that the number of data blocks n is equal to the number of leaf nodes in a full binary tree. Thus, for a file F = (m1, m2, m3, m4) where mı represents the ı-th block of the file, we can construct a tree as shown in Fig. 1a.Each node in HAT consists of a four-tuple νi =(i, li, vi, ti). i is the unique index of the node. The index of the root node is 1, and the indexes increases from top to bottom and from left to right. li denotes the number of leaf nodes that can be reached from the i-th node. Vi is the version number of the i-th node. ti represents the tag of the i-th node. When a HAT is initialized, the version number of each leaf is 1, and the version number of each non-leaf node is the sum of that of its two children. For the i-th node, mi denotes the combination of the blocks corresponding to its leaves. The tag ti is computed from F(mi), where F denotes a tag generation function. We require that for any node νi and its children ν2i and ν2i+1, F(mi) = F(m2i ⊙ m2i+1) = F(m2i) ⊗ F(m2i+1) holds, where ⊙ denotes the combination of m2i and m2i+1, and ⊗indicates the combination of F(m2i) and F(m2i+1), which is why we call it a "homomorphic" tree.

Pseudo code

4 THE CONSTRUCTION OF DEYPOS

In this section, we propose a concrete scheme of dedupli-catable dynamic PoS called DeyPoS. It consists of five algo-rithms as described in Section 2: Init, Encode, Deduplicate,Update, and Check.

4.1 Building Blocks We employ the following tools as our building blocks:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

1) Collision-resistant hash functions: A hash function
H : {0, 1}∗ → {0, 1}∗ is collision-resistant if the probability of finding two diffenent values x and y that satisfy H(x) = H(y) is negligible.

2) Deterministic symmetric encryption: The encryp-tion algorithm takes a key k and a plaintext m as input, and outputs the ciphertext. We use the nota-tion Enck(m) to denote the encryption algorithm.

3) Hash-based message authentication codes: A hash-based message authentication code
HMAC :{0, 1}∗ × {0, 1}∗ → {0, 1}∗ is a deterministic func-tion that takes a key k and an input x, and outputs a value y. We define HMACk(x) def = HMAC(k, x).

4) Pseudorandom functions: A pseudorandom func- tion f : {0, 1} ∗ ×{0, 1}∗ → {0, 1}∗ is a deterministic function that takes a key k and a value x, and outputs a value y that is indistinguishable from a truly random function of the same input x. We define fk(x) def = f(k, x).
5) Pseudorandom permutations: A pseudorandom permutation π : {0, 1} ∗ × [1, n] → [1, n] is a deter-ministic function that takes a key k and an integer x where 1 ⩽ x ⩽ n, and outputs a value y where 1 ⩽ y ⩽ n that is indistinguishable from a truly random permutation of the same input x. We define π k(x) def= π(k, x)

IV. .SIMULATION RESULTS

5 PERFORMANCE EVALUATION

In this section, we evaluate the performance of DeyPoS.The evaluation is divided into two aspects: theoretical and experimental evaluations.

5.1 Theoretical Comparison

Table 1 summarizes the asymptotic performance of our scheme in comparison with related schemes, where n de-notes the number of blocks, b denotes the number of the challenged blocks, and |m| denotes the size of one block. From the table, we observe that our scheme is the only one satisfying the cross-user deduplication on the client-side and dynamic proof of storage simultaneously. Furthermore, the asymptotic performance of our scheme is better than the other schemes except [31], which only provides weak security guarantee.

5.2 Experimental Comparison

We implemented our construction by OpenSSL 1.0.1 on a computer with an Intel 3.2GHz CPU and 8GB DDR4 memory. Each data point is the average of ten experiment results. The evaluation consists of three aspects, including the cost in the upload phase, the cost in the deduplication phase, and the cost in the proof of storage phase. The cost in the update phase is similar to the cost in the proof of storage phase, thus, we do not present the cost in the update phase. From Table 1, we know that the skip list is only used in dynamic PoS [8], while the Merkle tree is used in both dynamic PoS [14] and deduplication [15]. In addition, the theoretical performance of the skip list is similar to the Merkle tree. Hence, we only compare our scheme with the Merkle tree based solutions. Since there is no Merkle tree based solution that supports both dynamic PoS and deduplication, we compare our scheme with the one based on Merkle tree (like [14] [15]).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 9, September 2017

TABLE 1
Comparison with related schemes

Scheme	Deduplication on the client-side			Proof of Storage		
	Client	Server	Comm	Client	server	Comm
[15]	$O(n^2)$	$O(b \log n)$	$O(b \log n) + b\|m\|$	N/A		
[31]	$O(n)$	$O(1)$	$O(1)$	N/A		
[8]	N/A			$O(b \log n)$	$O(b \log n)$	$O(b \log n) + b\|m\|$
[14]	N/A			$O(b \log n)$	$O(b \log n)$	$O(b^2 \log n) + \|m\|$
This paper	$O(n)$	$O(b \log n)$	$O(b \log n) + \|m\|$	$O(b \log n)$	$O(b \log n)$	$O(b \log n) + \|m\|$

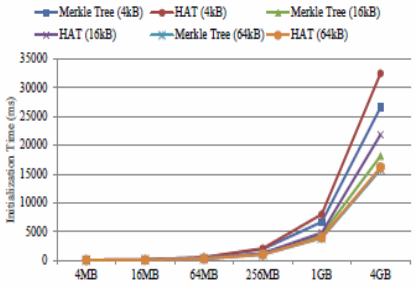


Fig. 3. Initialization time in different file sizes

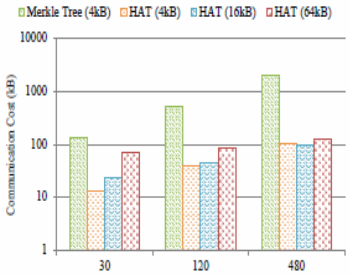


Fig. 5. Communication cost of 1GB file in the deduplication phase, when the number of challenged blocks are 30, 120, and 480, respectively

V. CONCLUSION AND FUTURE WORK

We proposed the comprehensive requirements in multi-user cloud storage systems and introduced the model of dedupli- catable dynamic PoS. We designed a novel tool called HAT which is an efficient authenticated structure. Based on HAT, we proposed the first practical deduplicatable dynamic PoS scheme called DeyPoS and proved its security in the ran-dom oracle model. The theoretical and experimental results show that our DeyPoS implementation is efficient, especially when the file size and the number of the challenged blocks are large.

REFERENCES

1. S. Kamara and K. Lauter, "Cryptographic cloud storage," in Proc.of FC, pp. 136–149, 2010.
2. G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," in Proc. of SecureComm, pp. 1–10, 2008.
3. Z. Xiao and Y. Xiao, "Security and privacy in cloud computing,"IEEE Communications Surveys Tutorials, vol. 15, no. 2, pp. 843–859, 2013.
4. A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in Proc. of CCS, pp. 584–597, 2007.
5. C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From Security to Assurance in the Cloud: A Survey," ACM Comput. Surv., vol. 48, no. 1, pp. 2:1–2:50, 2015.
6. C. Erway, A. K ¨upc ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. of CCS, pp. 213–222, 2009.

BIOGRAPHY

M. Mrudula is currently Pursuing M. Tech from Prakasam Engineering College, Prakasam, Andhra Pradesh. India