iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** XI **Month of publication:** November 2021

DOI: https://doi.org/10.22214/ijraset.2021.38797

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

Persistent Cohesion with Advanced Ring Signatures for Shared Data in Cloud

M. Shaheda Begum[1], S. Rumana Firdose[2]
[1, 2]*Assistant Professor in Computer Science and Engineering, Ashoka Women's Engineering College, A.P., India*

Abstract: ***Motivated by the exponential growth and the huge success of cloud data services bring the cloud common place for data to be not only stored in the cloud, but also shared across multiple users. Our scheme also has the added feature of access control in which only valid users are able to decrypt the stored information. Unfortunately, the integrity of cloud data is subject to skepticism due to the existence of hardware/software failures and human errors. Several mechanisms have been designed to allow both data owners and public verifiers to efficiently audit cloud data integrity without retrieving the entire data from the cloud server. However, public auditing on the integrity of shared data with these existing mechanisms will inevitably reveal confidential information—identity privacy—to public verifiers. In this paper, we propose a novel privacy-preserving mechanism that supports public auditing on shared data stored in the cloud. In particular, we exploit ring signatures to compute verification metadata needed to audit the correctness of shared data. With our mechanism, the identity of the signer on each block in shared data is kept private from public verifiers, who are able to efficiently verify shared data integrity without retrieving the entire file. In addition, our mechanism is able to perform multiple auditing tasks simultaneously instead of verifying them one by one. Our experimental results demonstrate the effectiveness and efficiency of our mechanism when auditing shared data integrity.***
Keywords: Public auditing, privacy-preserving, shared data, cloud computing

I. INTRODUCTION

Now a days Cloud service providers offer users efficient and scalable data storage services with a much lower marginal cost than traditional approaches [1]. It is routine for users to leverage cloud storage services to share data with others in a group, as data sharing becomes a standard feature in most cloud storage offerings, including Dropbox, iCloud and Google Drive.

The integrity of data in cloud storage, however, is subject to skepticism and scrutiny, as data stored in the cloud can easily be lost or corrupted due to the inevitable hardware/ software failures and human errors [3]. To make this matter even worse, cloud service providers may be reluctant to inform users about these data errors in order to maintain the reputation of their services and avoid losing profits [3]. Therefore, the integrity of cloud data should be verified before any data utilization, such as search or computation over cloud data [4].

The traditional approach for checking data correctness is to retrieve the entire data from the cloud, and then verify data integrity by checking the correctness of signatures (e.g., RSA [5]) or hash values (e.g., MD5 [6]) of the entire data. Certainly, this conventional approach is able to successfully check the correctness of cloud data. However the efficiency of using this traditional approach on cloud data is in doubt.

The main reason is that the size of cloud data is large in general. Downloading the entire cloud data to verify data integrity will cost or even waste user's amounts of computation and communication resources, especially when data have been corrupted in the cloud. Besides, many uses of cloud data (e.g., data mining and machine learning) do not necessarily need users to download the entire cloud data to local devices [2]. It is because cloud providers, such as Amazon, can offer users computation services directly on large-scale data that already existed in the cloud.

Recently, many mechanisms [7] have been proposed to allow not only a data owner itself but also a public verifier to efficiently perform integrity checking without downloading the entire data from the cloud, which is referred to as public auditing [5]. In these mechanisms, data is divided into many small blocks, where each block is independently signed by the owner; and a random combination of all the blocks instead of the whole data is retrieved during integrity checking [9]. A public verifier could be a data user (e.g., researcher) who would like to utilize the owner's data via the cloud or a third-party auditor (TPA) who can provide expert integrity checking services. Moving a step forward, Wang et al. designed an advanced auditing mechanism [5] (named as WWRL in this paper), so that during public auditing on cloud data, the content of private data belonging to a personal user is not disclosed to any public verifiers.

©IJRASET: All Rights are Reserved

Unfortunately, current public auditing solutions mentioned above only focus on personal data in the cloud.

We believe that sharing data among multiple users is perhaps one of the most engaging features that motivate cloud storage. Therefore, it is also necessary to ensure the integrity of shared data in the cloud is correct. Existing public auditing mechanisms can actually be extended to verify shared data integrity. However, a new significant privacy issue introduced in the case of shared data with the use of existing mechanisms is the leakage of identity privacy to public verifiers.



Figure 1 Alice and Bob share a data file in the cloud, and a public verifier audits shared data integrity with existing mechanisms.

For instance, Alice and Bob work together as a group and share a file in the cloud (as presented in Fig. 1). The shared file is divided into a number of small blocks, where each block is independently signed by one of the two users with existing public auditing solutions (e.g., [5]). Once a block in this shared file is modified by a user, this user needs to sign the new block using his/her private key. Eventually, different blocks are signed by different users due to the modification introduced by these two different users. Then, in order to correctly audit the integrity of the entire data, a public verifier needs to choose the appropriate public key for each block (e.g., a block signed by Alice can only be correctly verified by Alice's public key). As a result, this public verifier will inevitably learn the identity of the signer on each block due to the unique binding between an identity and a public key via digital certificates under public key infrastructure (PKI). Failing to preserve identity privacy on shared data during public auditing will reveal significant confidential information (e.g., which particular user in the group or special block in shared data is a more valuable target) to public verifiers. Specifically, as shown in Fig. 1, after performing several auditing tasks, this public verifier can first learn that Alice may be a more important role in the group because most of the blocks in the shared file are always signed by Alice; on the other hand, this public verifier can also easily deduce that the eighth block may contain data of a higher value (e.g., a final bid in an auction), because this block is frequently modified by the two different users. In order to protect this confidential information, it is essential and critical to preserve identity privacy from public verifiers during public auditing.

In this paper, to solve the above privacy issue on shared data, we propose Oruta,1 a novel privacy- preserving public auditing mechanism. More specifically, we utilize ring signatures to construct homomorphic authenticators in Oruta, so that a public verifier is able to verify the integrity of shared data without retrieving the entire data—while the identity of the signer on each block in shared data is kept private from the public verifier. In addition, we further extend our mechanism to support batch auditing, which can perform multiple auditing tasks simultaneously and improve the efficiency of verification for multiple auditing tasks. Meanwhile, Oruta is compatible with random masking [5], which has been utilized in WWRL and can preserve data privacy from public verifiers.

Moreover, we also leverage index hash tables from a previous public auditing solution [8] to support dynamic data.

II. PROBLEM STATEMENT

A. System Model

As illustrated in Fig. 2, the system model in this paper involves three parties: the cloud server, a group of users and a public verifier. There are two types of users in a group: the original user and a number of group users. The original user initially creates shared data in the cloud, and shares it with group users.

Both the original user and group users are members of the group. Every member of the group is allowed to access and modify shared data. Shared data and its verification metadata (i.e., signatures) are both stored in the cloud server. A public verifier, such as a third party auditor providing expert data auditing services or a data user outside the group intending to utilize shared data, is able to publicly verify the integrity of shared data stored in the cloud server

When a public verifier wishes to check the integrity of shared data, it first sends an auditing challenge to the cloud server. After receiving the auditing challenge, the cloud server responds to the public verifier with an auditing proof of the possession of shared data. Then, this public verifier checks the correctness of the entire data by verifying the correctness of the auditing proof. Essentially, the process of public auditing is a challenge and response protocol between a public verifier and the cloud server.

Valid verification metadata (i.e., signatures) on shared data. (4) Identity Privacy: A public verifier cannot distinguish the identity of the signer on each block in shared data during the process of auditing.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

1) Possible Alternative Approaches

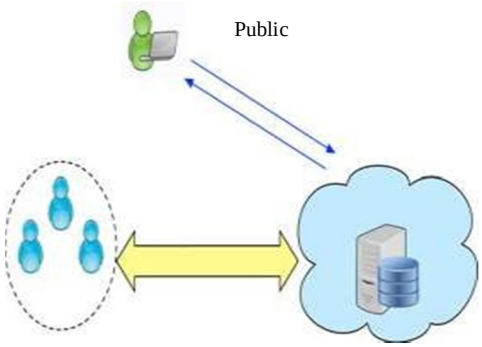


Figure 2 System Model

B. Threat Model

1) Integrity Threats. Two kinds of threats related to the integrity of shared data are possible. First, an adversary may try to corrupt the integrity of shared data. Second, the cloud service provider may inadvertently corrupt (or even remove) data in its storage due to hardware failures and human errors. Making matters worse, the cloud service provider is economically motivated, which means it may be reluctant to inform users about such corruption of data in order to save its reputation and avoid losing profits of its services.

2) Privacy Threats. The identity of the signer on each block in shared data is private and confidential to the group. During the process of auditing, a public verifier, who is only allowed to verify the correctness of shared data integrity, may try to reveal the identity of the signer on each block in shared data based on verification metadata. Once the public verifier reveals the identity of the signer on each block, it can easily distinguish a high-value target (a particular user in the group or a special block in shared data) from others.

C. Design Objectives

Our mechanism, Oruta, should be designed to achieve following properties: (1) Public Auditing: A public verifier is able to publicly verify the integrity of shared data without retrieving the entire data from the cloud. (2) Correctness: A public verifier is able to correctly verify shared data integrity. (3) Unforgeability: Only a user in the group can generate To preserve the identity of the signer on each block during public auditing, one possible alternative approach is to ask all the users of the group to share a global private key. Then, every user is able to sign blocks with this global private key. However, once one user of the group is compromised or leaving the group, a new global private key must be generated and securely shared among the rest of the group, which clearly introduces huge overhead to users in terms of key management and key distribution. While in our solution, each user in the rest of the group can still utilize its own private key for computing verification metadata without generating or sharing any new secret keys. Trusted Computing offers another possible alternative approach to achieve the design objectives of our mechanism. Specifically, by utilizing direct anonymous attestation, which is adopted by the Trusted Computing Group as the anonymous method for remote authentication in trusted platform module, users are able to preserve their identity privacy on shared data from a public verifier. The main problem with this approach is that it requires all the users using designed hardware, and needs the cloud provider to move all the existing cloud services to the trusted computing environment, which would be costly and impractical.

D. Ring Signatures

The concept of ring signatures was first proposed by Rivest et al. [8] in 2001. With ring signatures, a verifier is convinced that a signature is computed using one of group members' private keys, but the verifier is not able to determine which one. More concretely, given a ring signature and a group of d users, a verifier cannot distinguish the signer's identity with a probability more than 1=d. This property can be used to preserve the identity of the signer from a verifier.

The ring signature scheme introduced by Boneh et al. [9] (referred to as BGLS in this paper) is constructed on bilinear maps. We will extend this ring signature scheme to construct our public auditing mechanism.

©IJRASET: All Rights are Reserved

E. Homomorphic Authenticators

Homomorphic authenticators (also called homomorphic verifiable tags) are basic tools to construct public auditing mechanisms [10]. Besides unforgeability (i.e., only a user with a private key can generate valid signatures), a homomorphic authenticable signature scheme, which denotes a homomorphic authenticator based on signatures.

III. PUBLIC AUDITING MECHANISM

A. Overview

As we introduced in previous sections, we intend to uti-lize ring signatures to hide the identity of the signer on each block, so that private and sensitive information of the group is not disclosed to public verifiers. However, traditional ring signatures [2], [8] cannot be directly used into public auditing mechanisms, because these ring signature schemes do not support blockless verifiability. Without blockless verifiability, a public verifier has to download the whole data file to verify the correctness of shared data, which consumes excessive bandwidth and takes very long verification times. Therefore, we design a new homomorphic authentica-ble ring signature (HARS) scheme, which is extended from a classic ring signature scheme [21]. The ring signa-tures generated by HARS are not only able to preserve identity privacy but also able to support blockless verifi- ability. We will show how to build the privacy- preserv-ing public auditing mechanism for shared data in the cloud based on this new ring signature scheme in the next section.

Using HARS and its properties we established in the previous section, we now construct Oruta, a privacy-preserving public auditing mechanism for shared data in the cloud. With Oruta, the public verifier can verify the integrity of shared data without retrieving the entire data. Meanwhile, the identity of the signer on each block in shared data is kept private from the public verifier during the auditing.

B. Reduce Signature Storage

Another important issue we should consider in the construction of Oruta is the size of storage used for ring signatures. According to the generation of ring signatures in HARS, a block m is an element of Z_p and its ring signature contains d elements of G1, where G_1 is a cyclic group with order p. It means a |p|-bit block requires a d x |p|-bit ring signature, which forces users to spend a huge amount of space on storing ring signatures. It will be very frustrating for users, because cloud service providers, such as Amazon, will charge users based on the storage space they use.

C. Support Dynamic Operations

To enable each user in the group to easily modify data in the cloud, Oruta should also support dynamic operations on shared data. A dynamic operation includes an insert, delete or update operation on a single block. However, since the computation of a ring signature includes an identifier of a block (as presented in HARS), traditional methods, which only use the index of a block as its identifier (i.e., the index of block m_j is j), are not suitable for supporting dynamic operations on shared data efficiently.

D. New Ring Signature Scheme

As I am introduced in previous sections, we intend to uti-lize ring signatures to hide the identity of the signer on each block, so that private and sensitive information of the group is not disclosed to public verifiers. However, traditional ring signatures [1], [2] cannot be directly used into public auditing mechanisms, because these ring signature schemes do not support blockless verifiability. Without blockless verifiability, a public verifier has to download the whole data file to verify the correctness of shared data, which consumes excessive bandwidth and takes very long verification times. Here for, we design a new homomorphic authentica-ble ring signature (HARS) scheme, which is extended from a classic ring signature scheme [1]. The ring signa-tures generated by HARS are not only able to preserve identity privacy but also able to support block less verify- ability. We will show how to build the privacy-preserve- ing public auditing mechanism for shared data in the cloud based on this new ring signature scheme in the next section.

IV. RELATED WORK

Provable data possession (PDP), proposed by Ateniese et al. [11], allows a verifier to check the correctness of a client's data stored at an untrusted server. By utilizing RSA-based homomorphic authenticators and sampling strategies, the verifier is able to publicly audit the integrity of data without retrieving the entire data, which is referred to as public auditing. Unfortunately, their mechanism is only suitable for auditing the integrity of personal data. Juels and Kaliski defined another similar model called Proofs of Retrievability (POR), which is also able to check the correctness of data on an untrusted server.

©IJRASET: All Rights are Reserved

The original file is added with a set of randomly-valued check blocks called sentinels. The verifier challenges the untrusted server by specifying the positions of a collection of sentinels and asking the untrusted server to return the associated sentinel values. Shacham and Waters [10] designed two improved schemes. The first scheme is built from BLS signatures, and the second one is based on pseudo-random functions.

To support dynamic data, Ateniese et al. [12] presented an efficient PDP mechanism based on symmetric keys. This mechanism can support update and delete operations on data, however, insert operations are not available in this mechanism. Because it exploits symmetric keys to verify the integrity of data, it is not public verifiable and only provides a user with a limited number of verification requests. Wang et al. [12] utilized Merkle Hash Tree and BLS signatures to support dynamic data in a public auditing mechanism. Erway et al. [11] introduced dynamic provable data possession (DPDP) by using authenticated dictionaries, which are based on rank information. Zhu et al. [5] exploited the fragment structure to reduce the storage of signatures in their public auditing mechanism. In addition, they also used index hash tables to provide dynamic operations on data. The public mechanism proposed by Wang et al. [5] and its journal version [8] are able to preserve users' confidential data from a public verifier by using random maskings. In addition, to operate multiple auditing tasks from different users efficiently, they extended their mechanism to enable batch auditing by leveraging aggregate signatures.

V. CONCLUSION

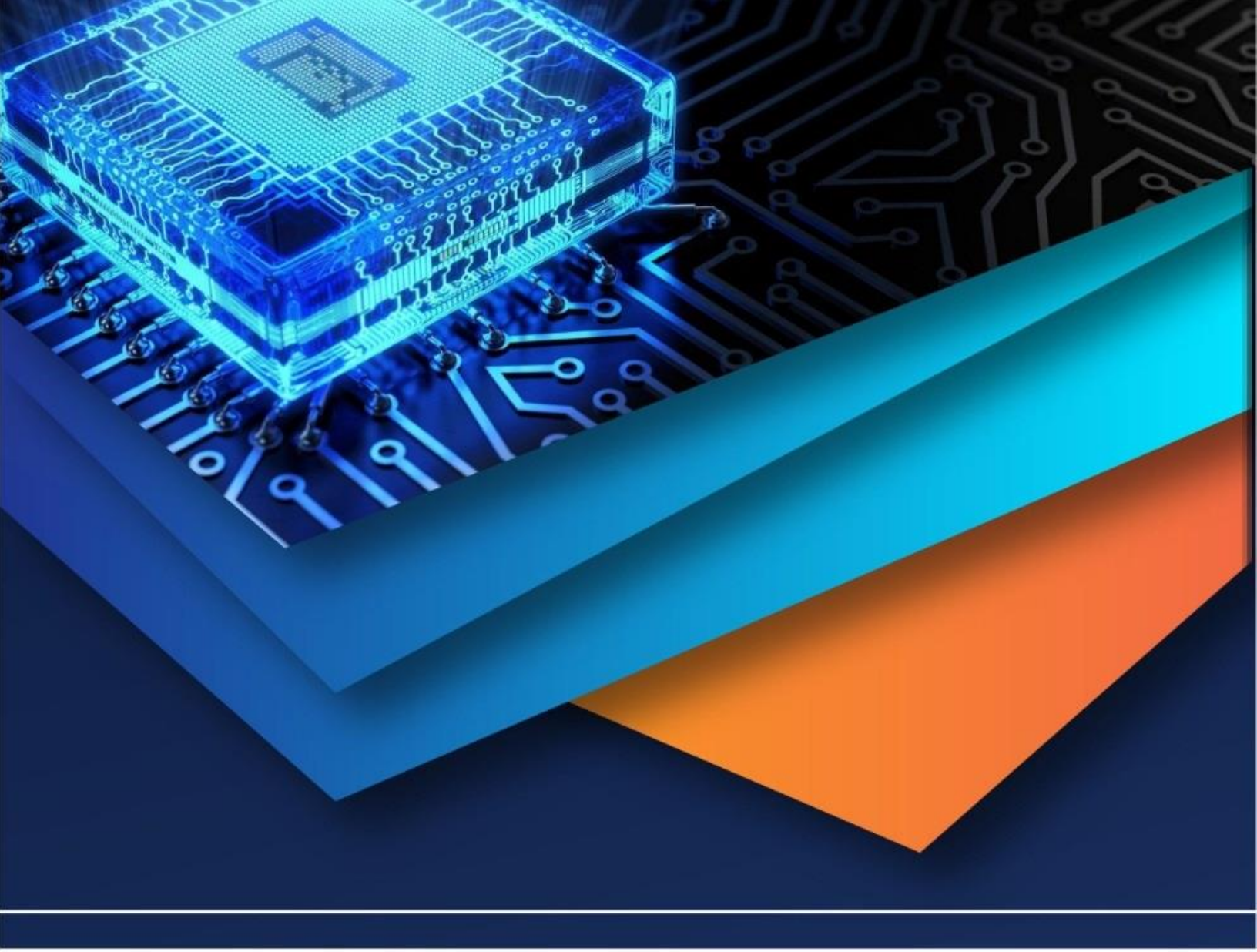
In this paper, we propose Oruta, a privacy- preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphic authenticators, so that a public verifier is able to audit shared data integrity without retrieving the entire data, yet it cannot distinguish who is the signer on each block. To improve the efficiency of verifying multiple auditing tasks, we further extend our mechanism to support batch auditing.

There are two interesting problems we will continue to study for our future work. One of them is traceability, which means the ability for the group manager (i.e., the original user) to reveal the identity of the signer based on verification metadata in some special situations. Since Oruta is based on ring signatures, where the identity of the signer is unconditionally protected [2], the current design of ours does not support traceability. To the best of our knowledge, designing an efficient public auditing mechanism with the capabilities of preserving identity privacy and supporting traceability is still open. Another problem for our future work is how to prove data freshness (prove the cloud possesses the latest version of shared data) while still preserving identity privacy.

REFERENCES

[1] 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, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.
[2] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.
[3] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533, 2010.
[4] B. Wang, M. Li, S.S. Chow, and H. Li, "Computing Encrypted Cloud Data Efficiently under Multiple Keys," Proc. IEEE Conf. Comm. and Network Security (CNS '13), pp. 90-99, 2013.
[5] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Comm. ACM, vol. 21, no. 2, pp. 120-126, 1978.
[6] The MD5 Message-Digest Algorithm (RFC1321). https://tools. ietf.org/html/rfc1321, 2014.
[7] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08), pp. 90- 107, 2008.
[8] R.L. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret," Proc. Seventh Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'01), pp. 552- 565, 2001.
[9] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'03), pp. 416-432, 2003.
[10] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S.S Yau, "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds," Proc. ACM Symp. Applied Computing (SAC'11), pp. 1550- 1557, 2011.
[11] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-610, 2007.
[12] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Networks (SecureComm'08), 2008.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)