ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud

Nalla Karthik Reddy
M.Tech Student,
Department of CSE,
AMR Institute of Technology,
Mavala, T.S, India.

Miss. Sara Anjum
Assistant Professor
Department of CSE,
AMR Institute of Technology,
Mavala, T.S, India.

Abstract—With cloud data services, it is commonplace for data to be not only stored in the cloud, but also shared across multiple users. 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.

Index Terms—Public auditing, privacy-preserving, shared data, cloud computing

1 INTRODUCTION

CLOUD service providers manage an enterprise-class infrastructure that offers a scalable, secure and reliable environment for users, at a much lower marginalcost due to the sharing nature of resources. It is routine for users to use cloud storage services to share data with others in a team, as data sharing becomes a standard feature in most cloud storage offerings, including Dropbox and Google Docs. The integrity of data in cloud storage, however, is subject to skepticism and scrutiny, as data stored in an untrusted cloud can easily be lost or corrupted, due to hardware failures and human errors [1]. To protect the integrity of cloud data, it is best to perform public auditing by introducing a third party auditor (TPA), who offers its auditing service with more powerful computation and communication abilities than regular users. The first provable data possession (PDP) mechanism [2] to perform public auditing is designed to check the correctness of data stored in an untrusted server, without retrieving the entire data. Moving a step forward, Wang *et al.* [3] (referred to as WWRL in this paper) is designed to construct a public auditing mechanism for cloud data, so that during public auditing, the content of private data belonging to a personal user is not disclosed to the third party auditor.

We believe that sharing data among multiple users is perhaps one of the most engaging features that motivates cloud storage. A unique problem introduced during the process of public auditing for shared data in the cloud is how to preserve **identity privacy** from the TPA, because the identities of signers on shared data may indicate that a particular user in the group or a special block in shared data is a higher valuable target than others. For example, Alice and Bob work together as a group and share a file in the cloud. The shared file is divided into a number of small blocks, which are independently signed by users. Once a block in this shared file is modified by a user, this user needs to sign the new block using her public/private key pair. The TPA needs to know the identity of the signer on each block in this shared file, so that it is able to audit the integrity of the whole file based on requests from Alice or Bob.

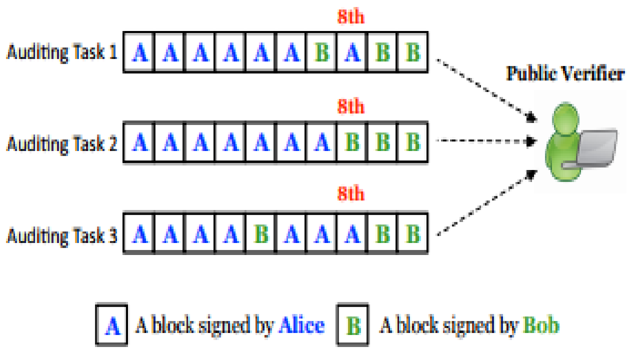

Fig. 1. Alice and Bob share a data file in the cloud, and a public verifier audits shared data integrity with existing mechanisms.

As shown in Fig. 1, after performing several auditing tasks, some private and sensitive information may reveal to the TPA. On one hand, most of the blocks in shared file are signed by Alice, which may indicate that Alice is a important role in this group, such as a group leader. On the other hand, the 8-th block is frequently modified by different users. It means this block may contain highvalue data, such as a final bid in an auction, that Alice and Bob need to discuss and change it several times. As described in the example above, the identities of signers on shared data may indicate which user in the group or block in shared data is a higher valuable target than others. Such information is confidential to the group and should not be revealed to any third party. However, no existing mechanism in the literature is able to perform public auditing on shared data in the cloud while still preserving identity privacy. In this paper, we propose Oruta1, a new privacy preserving public auditing mechanism for shared data in an untrusted cloud.

In Oruta, we utilize ring signatures [4], [5] to construct homomorphic authenticators [2], [6], so that the third party auditor is able to verify the integrity of shared data for a group of users without etrieving the entire data — while the identity of the signer on each block in shared data is kept private from the TPA. In addition, we further extend our mechanism to support batch auditing, which can audit multiple shared data simultaneously in a single auditing task. Meanwhile, Oruta continues to use random masking [3] to support data privacy during public auditing, and leverage index hash tables [7] to support fully dynamic operations on shared data. A dynamic operation indicates an insert, delete or update operation on a single block in shared data. A high-level comparison between Oruta and existing mechanisms in the literature is shown in Table 1. To our best knowledge, this paper represents the first attempt towards designing an effective privacy preserving public auditing mechanism for shared data.

TABLE 1
Comparison among Different Mechanisms

	PDP [9]	WWRL [5]	Oruta
Public Auditing	√	√	√
Data Privacy	×	√	√
Identity Privacy	×	×	√

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 [15] to support dynamic data.

A high-level comparison among Oruta and existing mechanisms is presented in Table 1. The remainder of this paper is organized as follows. In Section 2, we present the system model, threat model and design objectives. In Section 3, we introduce cryptographic primitives used in Oruta. The detailed design and security analysis of Oruta are presented in Section 4 and Section 5. In Section 6, we evaluate the performance of Oruta. Finally, we briefly discuss related work in Section 7, and conclude this paper in Section 8. Section 8.

2 PROBLEM STATEMENT

2.1 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 (TPA) 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 challengeand-response protocol between a public verifier and the cloud server [9].

2.2 Threat Model 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,

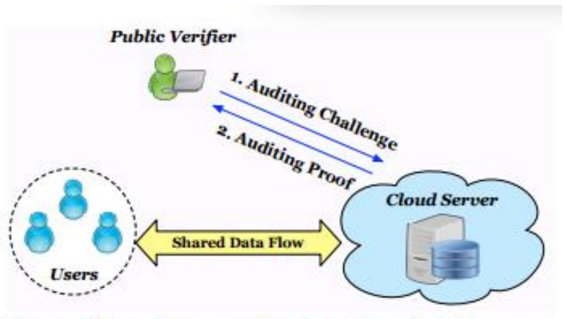

Fig. 2. Our system model includes the cloud server, a group of users and a public verifier.

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. 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 highvalue target (a particular user in the group or a special block in shared data) from others.

2.3 Design Objectives Our mechanism, Oruta, should be designed to achieve following properties: (1) Public Auditing: A public veri- fier 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 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.

2.4 Possible Alternative Approaches 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 [22], [23]. 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. Another possible approach to achieve identity privacy, is to add a trusted proxy between a group of users and the cloud in the system model. More concretely, each member's data is collected, signed, and uploaded to the cloud by this trusted proxy, then a public verifier can only verify and learn that it is the proxy signs the data, but cannot learn the identities of group members. Yet, the security of this method is threatened by the single point failure of the proxy. Besides, sometimes, not all the group members would like to trust the same proxy for generating signatures and uploading data on their behalf. Utilizing group signatures [24] is also an alternative option to preserve identity privacy. Unfortunately, as shown in our recent work [25], how to design an efficient public auditing mechanism based on group signatures remains open2 . Trusted Computing offers another possible alternative approach to achieve the design objectives of our mechanism. Specifically, by utilizing Direct Anonymous Attestation [26], 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.

3 PRELIMINARIES

In this section, we briefly introduce cryptographic primitives and their corresponding properties that we implement in Oruta.

3.1 Bilinear Maps

We first introduce a few concepts and properties related to bilinear maps. We follow notations from [5], [9]:

1) G_1, G_2 and G_T are three multiplicative cyclic groups of prime order p;
2) g_1 is a generator of G_1, and g_2 is a generator of G_2;
3) ψ is a computable isomorphism from G_2 to G_1, with $\psi(g_2) = g_1$;
4) e is a bilinear map e: $G_1 \times G_2 \rightarrow G_T$ with the following properties: **Computability**: there existsan efficiently computable algorithm for computing the map e. **Bilinearity**: for all $u \in G_1$, $v \in G_2$ and a, b $\in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$. **Non-degeneracy**:
$e(g_1, g_2) =6 1$.

These properties further imply two additional properties: (1) for any $u_1, u_2 \in G_1$ and $v \in G_2$, $e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v)$; (2) for any $u, v \in G_2$, $e(\psi(u), v) = e(\psi(v), u)$.

3.2 Complexity Assumptions

1: **Discrete Logarithm Problem.** For $a \in Z_p$, given g, $h = g^a \in G_1$, output a.

The Discrete Logarithm assumption holds in G_1 if no t-time algorithm has advantage at least ϙ in solving the Discrete Logarithm problem in G_1, which means it is computational infeasible to solve the Discrete Logarithm problem in G_1.

3.3 Ring Signatures

The concept of ring signatures is first proposed by Rivest *et al.* [4] 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. This property can be used to preserve the identity of the signer from a verifier. The ring signature scheme introduced by Boneh *et al.* [5] (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.

4 HOMOMORPHIC AUTHENTICABLE RING SIGNATURES

4.1 Overview

In this section, we introduce a new ring signature scheme, which is suitable for public auditing. Then, we will show how to build the privacy-preserving public auditing mechanism for shared data in the cloud based on this new ring signature scheme in the next section. As we introduced in previous sections, we intend to utilize 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 the TPA. However, traditional ring signatures [4], [5] cannot be directly used into public auditing mechanisms, because these ring signature schemes do not support blockless verification. Without blockless verification, the TPA has to download the whole data file to verify the correctness of shared data, which consumes excessive bandwidth and takes long verification times. Therefore, we first construct a new homomorphic authenticable ring signature (HARS) scheme, which is extended from a classic ring signature scheme [5], de-noted as BGLS. The ring signatures generated by HARS is able not only to preserve identity privacy but also to support blockless verification.

4.2 Construction of HARS

HARS contains three algorithms: **KeyGen**, **RingSign** and **RingVerify**. In **KeyGen**, each user in the group generates her public key and private key. In **RingSign**, a user in the group is able to sign a block with her private key and all the group members' public keys. A verifier is allowed to check whether a given block is signed by a group member in **RingVerify**. **Scheme Details.** Let G_1, G_2 and G_T be multiplicative cyclic groups of order p, g_1 and g_2 be generators of G_1 and G_2 respectively. Let $e : G_1 \times G_2 \to G_T$ be a bilinear map, and $\psi : G_2 \to G_1$ be a computable isomorphism with $\psi(g_2) = g_1$. There is a public map-to-point hash function H_1: $\{0, 1\}^* \to G_1$. The global parameters are $(e, \psi, p, G_1, G_2, G_T , g_1, g_2, H_1)$. The total number of users in the group is d. Let U denote the group that includes all the d users.

4.3 Security Analysis of HARS

Now, we discuss some important properties of HARS, including correctness, unforgeability, blockless verifica-tion, non-malleability and identity privacy.

***THEOREM** 1: Given any block and its ring signature, a verifier is able to correctly check the integrity of this block under HARS. Proof:* To prove the correctness of HARS is equiva-lent of proving Equation (3) is correct. Based on prop-erties of bilinear maps, the correctness of this equation can be proved as follows:

$$\begin{aligned}
\prod_{i=1}^{d} e(\sigma_i, w_i) &= e(\sigma_s, w_s) \cdot \prod_{i \neq s} e(\sigma_i, w_i) \\
&= e\left(\left(\frac{\beta}{\psi(\prod_{i \neq s} w_i^{a_i})}\right)^{\frac{1}{x_s}}, g_2^{x_s}\right) \cdot \prod_{i \neq s} e(g_1^{a_i}, g_2^{x_i}) \\
&= e\left(\frac{\beta}{\psi(\prod_{i \neq s} g_2^{x_i a_i})}, g_2\right) \cdot \prod_{i \neq s} e(g_1^{a_i x_i}, g_2) \\
&= e\left(\frac{\beta}{\prod_{i \neq s} g_1{}^{a_i x_i}}, g_2\right) \cdot e\left(\prod_{i \neq s} g_1^{a_i x_i}, g_2\right) \\
&= e\left(\frac{\beta}{\prod_{i \neq s} g_1{}^{a_i x_i}} \cdot \prod_{i \neq s} g_1{}^{a_i x_i}, g_2\right) \\
&= e(\beta, g_2).
\end{aligned}$$

5 PRIVACY-PRESERVING PUBLIC AUDITING FOR SHARED DATA IN THE CLOUD

5.1 Overview

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

5.2 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 Zp and its ring signature contains d elements of G1, where G1 is a cyclic group with order p. It means a |p|-bit block requires a d × |p|-bit ring signature, which forces users to spend a huge amount of space on storing ring signatures. It is very frustrating for users, because cloud service providers, such as Amazon, will charge users based on the storage space they used. To reduce the storage for ring signatures and still allow the TPA to audit shared data efficiently, we exploit an aggregated approach from [6]. Specifically, we aggregate a block mj = (mj,1, ...,mj,k) 2 Zk p in shared data as Qk l=1 _mj,l instead of computing gm 1 in Equation (1), where _1, ..., _k are random values of G1. With the aggregation, the length of a ring signature is only d/k of the length of a block. Similar methods to reduce the storage space of signatures can also be found in [7]. Generally, to obtain a smaller size of a ring signature than the size of a block, we choose k > d. As a trade-off, the communication cost will be increasing with an increase of k.

5.3 Support Dynamic Operations

To enable each user in the group to easily modify data in the cloud and share the latest version of data with the rest of the group, Oruta should also support dynamic operations on shared data. An 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, are not suitable for supporting dynamic operations on shared data. The reason is that,when a user modifies a single block in shared data by performing an insert or delete operation, the indices of blocks that after the modified block are all changed (as shown in Figure 3 and 4), and the changes of these indices require users to re-compute the signatures of these blocks, even though the content of these blocks are not modified.

6 PERFORMANCE

In this section, we first analysis the computation and communication costs of Oruta, and then evaluate the performance of Oruta in experiments.

6.1 Computation Cost

The main cryptographic operations used in Oruta in-clude multiplications, exponentiations, pairing and hash-ing operations. For simplicity, we omit additions in the following discussion, because they are much easier to be computed than the four types of operations mentioned above. During auditing, the TPA first generates some random values to construct the auditing message, which only in-troduces a small cost in computation. Then, after receiv-ing the auditing message, the cloud server needs to com-pute a proof $\{\lambda,\mu,\varphi, \{id_j\}_{j\in J}\}$. The computation cost of calculating a proof is about $(k + dc)Exp_{G1} + dcMul_{G1} + ckMul_{Zp} + kHash_{Zp}$, where Exp_{G1} denotes the cost of computing one exponentiation in G_1, Mul_{G1} denotes the cost of computing one multiplication in G_1, Mul_{Zp} and $Hash_{Zp}$ respectively denote the cost of computing one multiplcation and one hashing operation in Z_p. To check the correctness of the proof $\{\lambda,\mu,\varphi, \{id_j\}_{j\in J}\}$, the TPA verifies it based on Equation (6). The total cost of verifying the proof is $(2k + c)Exp_{G1} + (2k + c)Mul_{G1} + dMul_{GT} + cHash_{G1} + (d + 2)Pair_{G1,G2}$. We use $Pair_{G1\ ,G2}$ to denote the cost of computing one pairing operation in G_1 and G_2.

6.2 Communication Cost

The communication cost of Oruta is mainly intro-duced by two factors: the auditing message and the auditing proof.

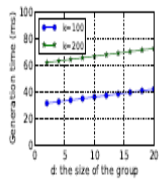

Fig. 10. Impact of d on signature generation time (ms).

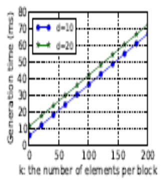

Fig. 11. Impact of k on signature generation time (ms).

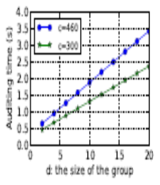

Fig. 12. Impact of d on auditing time (s), where $k = 100$.

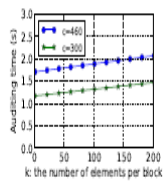

Fig. 13. Impact of k on auditing time (s), where $d = 10$.

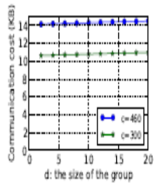

Fig. 14. Impact of d on communi-cation cost (KB), where $k = 100$.

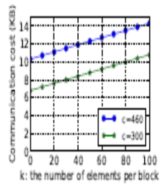

Fig. 15. Impact of k on communi-cation cost (KB), where $d = 10$.

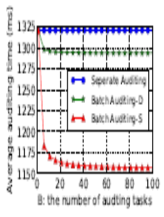

Fig. 16. Impact of B on the ef-ficiency of batch auditing, where $k = 100$ and $d = 10$.

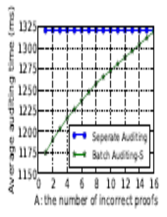

Fig. 17. Impact of A on the ef-ficiency of batch auditing, where $B = 128$.

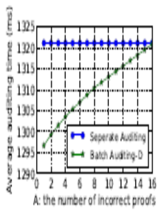

Fig. 18. Impact of A on the ef-ficiency of batch auditing, where $B = 128$.

For each auditing message $\{j, y_j\}_{j\in J}$, the communication cost is c(|q| + |n|) bits, where |q| is the length of an element of Zq and |n| is the length of an index. Each auditing = {_,μ,_, {idj}j2J } contains (k+d) elements of G1, k elements of Zp and c.

6.3 Experimental Results

We now evaluate the efficiency of Oruta in experiments.To implement these complex cryptographic operations that we mentioned before, we utilize the GNU Multiple Precision Arithmetic (GMP)2 library and Pairing Based Cryptography (PBC)3 library. All the following experiments are based on C and tested on a 2.26 GHz Linux system over 1, 000 times. Because Oruta needs more exponentiations than pairing operations during the process of auditing, the elliptic curve we choose in our experiments is an MNT curve with a base field size of 159 bits, which has a better performance than other curves on computing exponentiations. We choose |p| = 160 bits and |q| = 80 bits. We assume the total number of blocks in shared data is n = 1, 000, 000 and |n| = 20 bits. The size of shared data is 2 GB. To keep the detection probability greater than 99%, we set the number of selected blocks in an auditing task as c = 460 [2]. If only 300 blocks are selected, the detection probability is greater than 95%. We also assume the size of the group d 2 [2, 20] in the following experiments. Certainly, if a larger group size is used, the total computation cost will increase due to the increasing number of exponentiations and pairing.

7 RELATED WORK

Provable data possession (PDP), first proposed by Ateniese *et al.* [2], allows a verifier to check the correct-ness 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 verifiability or public auditing. Unfortunately, their mechanism is only suitable for auditing the integrity of static data. Juels and Kaliski [13] defined another similar model called proofs of re-trievability (POR), which is also able to check the correct-ness of data on an untrusted server. 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 [6] designed two improved POR schemes. The first scheme is built from BLS signatures, and the second one is based on pseudo-random functions. To support dynamic operations on data, Ateniese *et al.* [14] presented an efficient PDP mechanism based on symmetric keys.

This mechanism can support update and delete operations on data, however, insert opera-tions 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.* uti-lized Merkle Hash Tree and BLS signatures [9] to support fully dynamic operations in a public auditing mecha-nism. Erway *et al.* [15] introduced dynamic provable data possession (DPDP) by using authenticated dictionaries,which are based on rank information. Zhu *et al.* exploited the fragment structure to reduce the storage of signa-tures in their public auditing mechanism. In addition, they also used index hash tables to provide dynamic operations for users.

The public mechanism proposed by Wang *et al.* [3] is able to preserve users' confidential data from the TPA 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 [5]. Wang *et al.* [16] leveraged homomorphic tokens to ensure the correctness of erasure codes-based data dis-tributed on multiple servers. This mechanism is able not only to support dynamic operations on data, but also to identify misbehaved servers.

To minimize com-munication overhead in the phase of data repair, Chen *et al.* [17] also introduced a mechanism for auditing the correctness of data with the multi-server scenario, where these data are encoded by network coding instead of using erasure codes. More recently, Cao *et al.* [18] constructed an LT codes-based secure and reliable cloud storage mechanism. Compare to previous work [16], [17], this mechanism can avoid high decoding computation cost for data users and save computation resource for online data owners during data repair. To prevent special attacks exist in remote data storage system with *deduplication*, Halevi *et al.* [19] introduced the notation of proofs-of-ownership (POWs), which al-lows a client to prove to a server that she actually holds a data file, rather than just some hash values of the data file. Zheng *et al.* [20] further discussed that POW and PDP can co-exist under the same framework. Recently, Franz *et al.* [21] proposed an oblivious out-sourced storage scheme based on Oblivious RAM tech-niques, which is able to hide users' access patterns on outsourced data from an untrusted cloud. Vimercati *et al.* [22] utilize shuffle index structure to protect users' access patterns on outsourced data.

8 CONCLUSION

In this paper, we propose Oruta, the first privacy-preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphic authenticators, so the TPA is able to audit the integrity of shared data, yet cannot distinguish who is the signer on each block, which can achieve identity privacy. To improve the efficiency of verification for mul-tiple auditing tasks, we further extend our mechanism to support batch auditing. An interesting problem in our future work is how to efficiently audit the integrity of shared data with dynamic groups while still preserving the identity of the signer on each block from the third party auditor.

9.ACKNOWLEDGEMENTS

I am **NALLA KARTHIK REDDY** and would like to thank the publishers, researchers for making their resources material available. I am greatly thankful to Assistant Prof: **Miss. SARA ANJUM** for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

10.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," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apirl 2010.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2007, pp. 598–610.

[3] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 525–533.

[4] R. L. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret," in *Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer- Verlag, 2001, pp. 552–565.

[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in *Proc. In- ternational Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer-Verlag, 2003, pp. 416–432.

[6] H. Shacham and B. Waters, "Compact Proofs of Retrievability," in *Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer- Verlag, 2008, pp. 90–107.

[7] Y. Zhu, H.Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S.Yau, "Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds," in *Proc. ACM Symposium on Applied Computing (SAC)*, 2011, pp. 1550–1557.

[8] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 534–542.

Mr. NALLA KARTHIK REDDY. MTech student, inM.Tech Student, Dept of CSE in **Amr Institute of Technology**, mavala, T.S, India

Miss. SARA ANJUM working as a Assistant Professor at **Amr Institute of Technology**, mavala, T.S, India, Graduate from JNTUH Hyderabad. She has 2 years of UG/PG Teaching Experience