© 2015 IJIRT | Volume 1 Issue 12 | ISSN: 2349-6002

A Secure and Efficient Audit Mechanism for Dynamic Shared Data in Cloud Storage: ORUTA

J.Ranjitha[1], A.Aarthi[2]
[1]*Ponnaiyah Ramajeyam Engineering College, Vallam, Thanjavur*
[2]*Assistant Professor, Ponnaiyah Ramajeyam Engineering College, Thanjavur*

***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.

I INTRODUCTION

CLOUD service providers offer users efficient and scalable data storage services with a much lower marginal cost than traditional approaches. 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, I Cloud 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. 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. Therefore, the integrity of cloud data should be verified before any data utilization, such as search or computation over cloud data.

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) or hash values (e.g., MD5) 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. 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 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. 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. 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 (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 dis-closed to any public verifiers. Unfortunately, current public auditing solutions mentioned above only focus on personal data in the cloud.

II EASE OF USE

A Ensure the integrity of shared data

We believe that sharing data among multiple users is perhaps one of the most engaging features that motivates cloud storage. Therefore, it is also necessary to ensure the integrity of shared data in the cloud is correct. Exist 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 leak-age of identity privacy to public verifiers

B 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. 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 these confidential information, it is essential and critical to preserve identity privacy from public verifiers during public auditing.

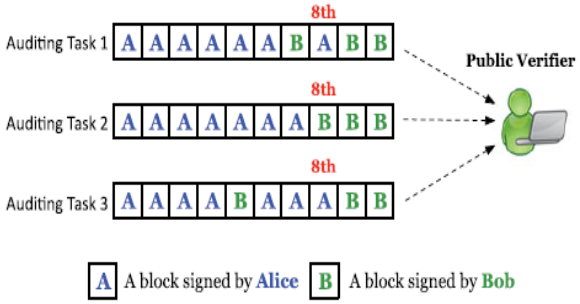


Fig. 1. Alice and Bob share a data file in the cloud, and a public verifier audits shared data integrity with existing mechanisms

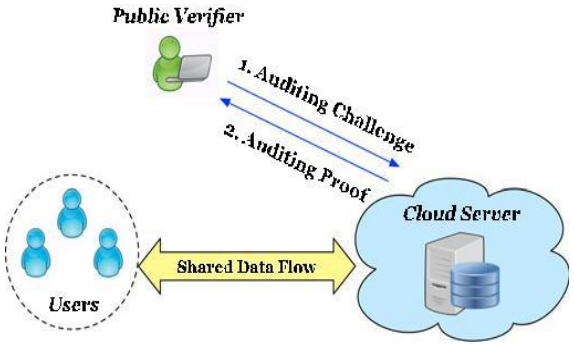
C ORUTA One Ring TO Rull Them ALL

In this paper, to solve the above privacy issue on shared data, we propose Oruta, 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 on each block in shared data is kept private from the public verifier. In without retrieving the entire data—while the identity of the signer

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, 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 to support dynamic data. A high-level comparison among Oruta and existing mechanisms.

In we present the system model, threat model and design objectives. we introduce cryptographic primitives used in Oruta. The detailed design and security analysis of Oruta are presented.

III ORUTA MECHANISMS



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

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, 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.

A NEW RING SIGNATURE SCHEME

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 public verifiers. However, traditional ring signatures cannot be directly used into public auditing mechanisms, because these ring signature schemes do not support block less verifiability. Without block less 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 Authenticable Ring Signature (HARS) scheme, which is extended from a classic ring signature scheme . The ring signatures generated by HARS are not only able to preserve identity privacy but also able to support block less verifiability. 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.

B Construction of HARS

HARS contains three algorithms: KeyGen, RingSign and RingVerify.

- In KeyGen, each user in the group generates his/her public key and private key.
- In RingSign, a user in the group is able to generate a signature on a block and its block identifier with his/her private key and all the group members' public keys.
- A block identifier is a string that can distinguish the corresponding block from others.
- A verifier is able to check whether a given block is signed by a group member in RingVerify

C PUBLIC AUDITING MECHANISMS

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. Mean while the identity of the signer on each block in shared data is kept private from the public verifier during the auditing.

Reducing 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 G_1, where G_1 is a cyclic group with order p. It means a bit block requires a bit ring sig-nature, 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.

To reduce the storage of ring signatures on shared data and still allow the public verifier to audit shared data efficiently, we exploit an aggregated approach from to expand the size of each block in shared data.

D CONSTRUCTION OF ORUTA

It includes five algorithms: Key Gen, Sig Gen, Modify, Proof Gen and Proof Verify.

- In Key Gen, users generate their own public/private key pairs.
- In Sig Gen, a user (either the original user or a group user) is able to compute ring signatures on blocks in shared data by using its own private key and all the group members' public keys.
- Each user in the group is able to perform an insert, delete or update operation on a block, and compute the new ring signature on this new block in Modify.
- Proof Gen is operated by a public verifier and the cloud server together to interactively generate a proof of possession of shared data.
- In Proof Verify, the public verifier audits the integrity of shared data by verifying the proof.

Note that for the ease of understanding, we first assume the group is static, which means the group is pre-defined before shared data is created in the cloud and the member-ship of the group is not changed during data sharing. Specifically, before the original user outsources shared data to the cloud, he/she decides all the group members. We will discuss the case of dynamic groups later.

In the construction of Oruta, we support data privacy by leveraging random masking (i.e.,Proof Gen), which is also used in previous work to protect data privacy for personal users. If a user wants to protect the content of private data in the cloud, this user can also encrypt data before outsourcing it into the cloud server with encryption techniques, such as the combination of symmetric key encryption and attribute-based encryption (ABE).

With the sampling strategy, which is widely used in most of the public auditing mechanisms, a public verifier can detect any corrupted block in shared data with a high probability by only choosing a subset of all blocks (i.e., choosing c-element subset from set in each auditing task., a public verifier can detect these corrupted blocks with a probability greater than 99 percent by choosing only 460 random blocks. Of course, this public verifier can always spend more communication overhead, and verify the integrity of data by choosing all the n blocks in shared data. Even if all the n blocks in shared data are selected (i.e., with-out using sampling strategy), the communication over-head during public auditing is still much more smaller than retrieving the entire data from the cloud.

.

Besides choosing a larger number of random blocks, another possible approach to improve the detection probability is to perform multiple auditing tasks on the same shared data by using different random (i.e., is different for blocking each different task). Specifically, if the cur-rent detection probability is and a number of auditing tasks is performed, then the detection probability is computed.

Dynamic Groups:. If a new user can be added in the group or an existing user can be revoked from the group, then this group is denoted as a dynamic group. To support dynamic groups while still allowing the public verifier to perform public auditing, all the ring signatures on shared data need to be re-computed with the signer's private key and all the current users' public keys when the member-ship of the group is changed.

For example, if the current size of the group is d and a new user is added into the group, then a ring signature on each block in shared data needs to be re-computed with the signer's private key and all the public keys. If the current size of the group is d and an existing user u_d is revoked from the group, then a ring sig-nature on each block in shared data needs to be re-computed with the signer's private key and all the public keys.The main reason of this type of re-computation on signatures introduced by dynamic groups, is because the generation of a ring signature under our mechanism requires the signer's private key and all the current members' public keys. An interesting problem for our future work will be how to avoid this type of re-computation introduced by dynamic groups while still preserving identity privacy from the public verifier during the process of public auditing on shared data.

A Equations

Sometimes, a public verifier may need to verify the correctness of multiple auditing tasks in a very short time. Directly verifying these multiple auditing tasks separately would be inefficient. By leveraging the properties of bilinear maps, we can further extend Oruta to support batch auditing, which can verify the correctness of multiple auditing tasks simultaneously and improve the efficiency of public auditing.

Based on the correctness of the correctness of batch auditing in can be presented

$$\left(\prod_{b=1}^{B}\prod_{i=1}^{d_b} e(\phi_{b,i}, w_{b,i})\right)\cdot e\left(\prod_{b=1}^{B}\prod_{l=1}^{k}\lambda_{b,l}^{h(\lambda_{b,l})}, g_2\right)$$
$$=\prod_{b=1}^{B}\left(\left(\prod_{i=1}^{d_b} e(\phi_{b,i}, w_{b,i})\right)\cdot e\left(\prod_{l=1}^{k}\lambda_{b,l}^{h(\lambda_{b,l})}, g_2\right)\right)$$
$$=\prod_{b=1}^{B} e\left(\prod_{j\in\mathcal{J}} H(id_{b,j})^{y_j}\cdot\prod_{l=1}^{k}\eta_{b,l}^{\mu_{b,l}}, g_2\right)$$
$$=e\left(\prod_{b=1}^{B}\left(\prod_{j\in\mathcal{J}} H(id_{b,j})^{y_j}\cdot\prod_{l=1}^{k}\eta_{b,l}^{\mu_{b,l}}\right), g_2\right).$$

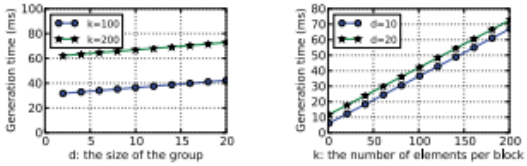
Which can save the public verifier about pairing operations in total compared batch auditing will fail if at least one incorrect auditing proof exists in all the B auditing proofs. To allow most of auditing proofs to still pass the verification when there exists only a small number of incorrect auditing proofs,

We can utilize binary search during batch auditing. More specifically, once the batch auditing of the B auditing proofs fails the public verifier divides the set of all the B auditing proofs into two subsets, where each subset contains a number of B=2auditing proofs. Then the public verifier re-checks the correctness of auditing proofs in each subset using batch auditing. If the verification result of one subset is correct, then all the auditing proofs in this subset are all correct. Otherwise, this subset is further divided into two sub-subsets, and the public verifier re checks the correctness of auditing proofs in each sub-subset with batch auditing until all the incorrect auditing proofs are found. Clearly, when the number of incorrect auditing proofs increases, the public verifier needs more time to distinguish all the incorrect auditing proofs, and the efficiency of batch auditing will be reduced. Experimental results in Section 6 show that, when less than 12 percent of all the B auditing proofs are incorrect, batching auditing is still more efficient than verifying all the B auditing proofs one by one.

B Figures

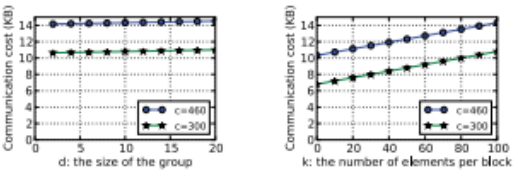
Utilization of GNU Multiple Precision Arithmetic (GMP) library and Pairing Based Cryptography (PBC) library. All the following experiments are based on C and tested on a 2.26 GHz Linux system over 1000 times. Because Oruta needs more exponentiations than pairing operations during the process of auditing, 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

160 bits and 80 bits. We assume the total number of blocks in shared data is n ¼ 1;000; 000 and 20 bits. The size of shared data is 2 GB. To keep the detection probability greater than 99 percent, we set the number of selected



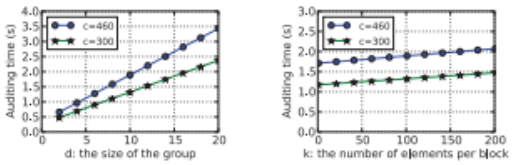
(a) Impact of d on signature generation time (ms). (b) Impact of k on signature generation time (ms).

Blocks in an auditing task as c ¼ 460. If only 300 blocks are selected, the detection probability is greater than 95 percent. We also assume the size of the group 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 operations. By this mechanism there is a way for hiding new signers from public verifier. While Verifier not only check the correctness of data but also try to make known who are all new identities to all. By allowing some special blocks consider these blocks are linear combination of all the process is accomplished.And also it is not a place for storing the data but also share across the multiple users.



(a) Impact of d on communication cost (KB), where $k = 100$. (b) Impact of k on communication cost (KB), where $d = 10$.

Performance of Signature Generation. According to Section 5, the generation time of a ring signature on a block is determined by the number of users in the group and the number of elements in each block. As illustrated in Figs. 10a and 10b, when k is fixed, the generation time of a ring signature is linearly increasing with the size of the group; when d is fixed, the generation time of a ring signature is linearly increasing with the number of elements in each block. Specifically, when d¼10 and k¼100, a user in the group requires about 37 milliseconds to compute a ring signature on a block in shared data.



(a) Impact of d on auditing time (second), where $k = 100$. (b) Impact of k on auditing time (second), where $d = 10$.

Performance of Batch Auditing. As in Oruta Mechanisms when there are multiple auditing proofs, the public verifier can improve the efficiency of verification by per-forming batch auditing.

IV CONCLUSION

In this paper 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, 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] B. Wang, B. Li, and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," Proc. IEEE Fifth Int'l Conf. Cloud Computing, pp. 295-302, 2012.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[3] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Pub-lic Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.

[4] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud Data Protection for the Masses," Computer, vol. 45, no. 1, pp. 39-45, 2012.

[5] 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.

[6] 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.

[7] H. Shacham and B. Waters, "Compact Proofs of
Proc. 14th Int'l Conf. Theory and Application of Cryptology
andInfor-mation Security.

[8] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Com-puter and Comm. Security (CCS'09), pp. 213-222, 2009.

[9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Secu-rity (ESORICS'09), pp. 355-370, 2009.

[10] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS'09), pp. 1-9, 2009.

[11] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote Data Checking for Network Coding-Based Distributed Storage Sys-tems," Proc. ACM Workshop Cloud Computing Security Workshop (CCSW'10), pp. 31-42, 2010.

[12] 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.

[13] N. Cao, S. Yu, Z. Yang, W. Lou, and Y.T. Hou, "LT Codes-Based Secure and Reliable Cloud Storage Service," Proc. IEEE INFO-COM, 2012.

[14] B. Wang, B. Li, and H. Li, "Certificateless Public Auditing for Data Integrity in the Cloud," Proc. IEEE Conf. Comm. and Network Security (CNS'13), pp. 276-284, 2013.

[15] C. Wang, S.S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Pre-serving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, vol. 62, no. 2, pp. 362-375, Feb. 2013.

[16] B. Wang, B. Li, and H. Li, "Public Auditing for Shared Data with Efficient User Revocation in the Cloud," Proc. IEEE INFOCOM, pp. 2904-2912, 2013.

[17] B. Wang, B. Li, and H. Li, "Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud," IEEE Trans. Services Computing, 20 Dec. 2013, DOI: 10.1109/TSC.2013.2295611.