Privacy of Shared Data with Efficient User Revocation for Regenerating Code Based Cloud using Predicate Encryption with Partial Key

Mrs. P. Sahila[1] PG scholar,
Department of Computer Science and Engineering SBM College of Engineering and Technology Dindigul, India
p.sahila1987@gmail.com

Mr.P.Rajapandi[2] Assistant Professor,
Department of Computer Science and Engineering
SBM College of Engineering and Technology
Dindigul, India
Rajapandi.dgl@gmail.com

***Abstract*—With data storage and sharing services in the cloud, end users can easily change and share file/data as a group or team. To make sure shared data integrity can be certified publicly, users in the group need to compute signatures on all the blocks/chunks in shared data. Different blocks/chunks in shared data are generally signed by different users due to data modifications made by different users. For security constrains, once a user is revoked from the group or team, the blocks or chunks which were previously signed by this revoked user must be re-signed by an existing user. The straightforward method is which allows an existing user to download the corresponding part of shared data and re-sign it during user revocation, is inefficient due to the huge size of shared data in the cloud. This paper proposes a novel public auditing technique for the integrity of shared files/data with efficient user revocation. By utilizing the idea of cloud service re-signatures, this system allows the cloud to re-sign blocks on behalf of proxy during user revocation, so that proxy does not need to download and re-sign blocks. In addition, a public verifier is always able to audit the integrity of shared files without downloading the entire files from the cloud, even if some part of shared data has been re-signed by the cloud. Thus, this scheme can completely release data owners from online load.**

***Index Terms*—Cloud storage, regenerating codes, public audit, privacy preserving, authenticator regeneration, re-signature services, privileged, and user revocation.**

I. INTRODUCTION

CLOUD storage is now attaining popularity because it provides an on-demand data outsourcing service with appealing benefits: release of the burden for storage management, widespread data access with location independence, and prevention of capital spending on hardware, software, and personal maintenances, etc., [1].

It is noted that data owners mislay ultimate control over the fate of their outsourced data; thus, the correctness, accessibility and integrity of the data are being put at risk. On the one hand, the cloud service is usually faced with a broad range of internal/external adversaries, who would maliciously delete or corrupt users' data; on the other hand, the cloud service providers may act dishonestly, attempting to hide data loss or corruption and claiming that the files are still correctly stored in the cloud for reputation or monetary reasons. Thus it makes great sense for users to implement an efficient protocol to perform periodical verifications of their outsourced data to ensure that the cloud indeed maintains their data correctly. Many mechanisms dealing with the integrity of outsourced data without a local copy have been proposed under different system and security models up to now. The most significant work among these studies are the PDP (*provable data possession*) model and POR (*proof of retrievability*) model, which were originally proposed for the **single-server** scenario by Ateniese *et al.* [2] and Juels and Kaliski [3], respectively. Considering that files are usually striped and redundantly stored across multi-servers or multi-clouds, [4]–[10] explore integrity verification schemes suitable for such **multi-servers** or **multi-clouds** setting with different redundancy schemes, such as *replication*, *erasure codes*, and, more recently, *regenerating codes*.

In this paper, we focus on the integrity verification problem in **regenerating-code-based cloud storage**, especially with the functional repair strategy [11]. Similar studies have been performed by Chen *et al.* [7] and Chen and Lee [8] separately and independently. [7] extended the single-server CPOR scheme (private version in [12]) to the regenerating code-scenario; [8] designed and implemented a data integrity protection (DIP) scheme for FMSR [13]-based cloud storage and the scheme is adapted to the thin-cloud setting.1 However, both of them are designed for private audit, only the data owner is allowed to verify the integrity and repair the faulty servers. Considering the large size of the outsourced data and the user's constrained resource capability, the tasks of auditing and reparation in the cloud can be formidable and expensive for the users [14]. The overhead of using cloud storage should be minimized as much as possible such that a user does not need to perform too many operations to their outsourced data (in additional to retrieving it) [15]. In particular, users may not want to go through the complexity in verifying and reparation. The auditing schemes in [7] and [8] imply the problem that users need to always stay online, which may impede its adoption in practice, especially for long-term archival storage.

To fully ensure the data integrity and save the users' computation resources as well as online burden, we propose a novel public auditing scheme for the regenerating-code-based cloud storage, in which the integrity checking and regeneration (of failed data blocks and authenticators) are implemented by a third-party auditor and a cloud re-signature services separately on behalf of the proxy. Instead of directly adapting the existing public auditing scheme [12] to the multi-server setting, we design a novel authenticator, which is more appropriate for regenerating codes. Besides, we —*encrypt* the coefficients to protect data privacy against the auditor, which is more lightweight than applying the proof blind technique in [14] and [15] and data blind method in [16]. Specifically, our contribution can be summarized by the following aspects:

- We design a novel homomorphic authenticator based on BLS signature [17], which can be generated by a couple of secret keys and verified publicly. Besides, it can be adapted for data owners equipped with low end computation devices in which they only need to sign the native blocks.
- To the best of our knowledge, our scheme is the first to allow privacy-preserving public auditing for regenerating code-based cloud storage. The coefficients are masked by a PRF (Pseudorandom Function) during the Setup phase to avoid leakage of the original data. This method is lightweight and does not introduce any computational overhead to the cloud servers or TPA.
- Our scheme completely releases data owners from online burden for the regeneration of blocks and authenticators at faulty servers and it provides the privilege to a re-signature service for the reparation.
- Optimization measures are taken to improve the flexibility and efficiency of our auditing scheme; thus, the storage overhead of servers, the computational overhead of the data owner and communication overhead during the audit phase can be effectively reduced.
- Our scheme is provable secure under *random oracle*. Moreover, we make a comparison with the state of the art and experimentally evaluate the performance of our scheme.

The rest of this paper is organized as follows: Section II introduces some preliminaries, the system model, threat model, design goals and formal definition of our auditing scheme.

II. SYSTEM MODEL AND PROBLEM STATEMENT

A. System Model

The system model for Regenerating-Code-based cloud storage as Fig.1, which involves four entities: *the data owner*, who owns large amounts of data files to be stored in the cloud; *the cloud*, which are managed by the cloud service provider, that provide storage service; *the third party auditor* (TPA),who has knowledge and capabilities to conduct public audits on the coded data/file in the cloud, the TPA is trusted and its audit results is unbiased for both data owners and cloud servers; and a *re-signature services* which regenerates authenticators and data blocks/chunks on the failed servers during the repair. In this the data owner is controlled in computational and storage resources compared to other objects and may become off-line even after the data upload procedure. To save resources as well as the online burden potentially brought by the periodic auditing and accidental repairing, the data owners resort to the TPA for integrity verification and delegate the reparation to the cloud re-generating services.

The cloud Re-generating services regenerates the authenticators without downloading the revoked user's data by using predicate partial key of the user. The privacy of outsourced data/file is maintained well and also they can overcome the security issues. The effectiveness of user revocation and computation and communication resources of existing users can be simply saved. This system achieves batch auditing where several outsourced auditing jobs from

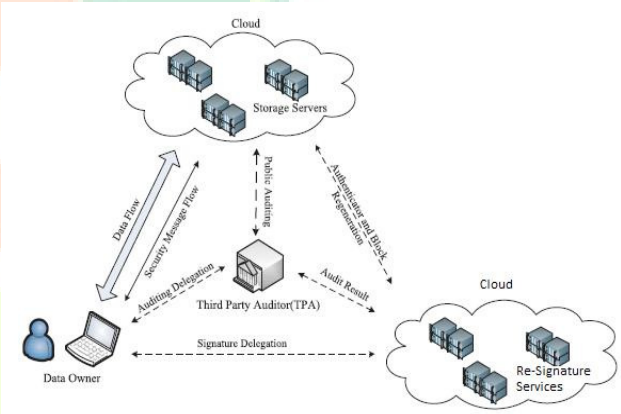

Fig. 1. The system model.

different users can be done concurrently by the TPA.

B. Problem Statement

To fully guarantee the data reliability and save the users computation resources as well as online load, this paper propose a public auditing system for the redeveloping-code-based cloud storage, in which the integrity checking and regeneration (of failed data/file blocks and authenticators) are implemented by a third-party auditor and a cloud re- signature services separately on behalf of the proxy. Instead of directly adapting the existing public auditing scheme to the multi-server setting, they design a novel public authenticator, which is more appropriate for regenerating codes. Besides, they —*encrypt* the coefficients to look after data privacy against the auditor.In addition, a public verifier is always able to audit the integrity of shared data without recovering the entire data from the cloud, even if some part of shared data has been re-signed by the cloud. Thus, this scheme can completely release data owners from online burden.

Cloud Computing makes these advantages more likeable than ever, it also brings new and exciting security threats towards user's outsourced data/file. Since cloud service providers (CSP) are distinct administrative units, data outsourcing is actually resigning user's ultimate switch over

the chance of their data. As a result, the accuracy of the data in the cloud is being set at threat due to the subsequent reasons. As users no longer substantially retain the storage of their file, outdated cryptographic primitives for the determination of data security protection cannot be directly accepted. Thus, how to efficiently confirm the exactness of outsourced cloud files/data without the local copy of data files becomes a big contest for data storage safety in Cloud Computing. Note that merely downloading the data/file for its integrity authentication is not a practical result due to the extravagance in I/O cost and transmitting the file/data through the network. Besides, it is repeatedly insufficient to identify the data corruption when retrieving the data/file, as it might be too late for progress the data/file loss or injury. Allowing for the huge size of the outsourced data/file and the user's controlled resource capability the skill to audit the accuracy of the data/file in a cloud site can be difficult and expensive for the cloud users. Therefore, to entirely ensure the data security and protect the cloud users' computation resources, it is of dangerous importance to allow public auditability for cloud records storage so that the user's may recourse to a third party auditor (TPA), who has expertise and competencies that the users do not, to audit the outsourced data/file when required. Based on the audit outcome, TPA could issue an audit report, which would not only support users to estimate the risk of their contributed cloud data service area, but also be valuable for the cloud service provider to develop their cloud built service platform .In a word, allowing public threat auditing practices will play an significant role for this developing cloud economy to become fully recognized, where users will need ways to evaluate risk and gain faith in Cloud.

The system can be précised as the following three aspects:

- This paper suggests a novel public auditing mechanism by using the idea of cloud service re-signatures; they permit the cloud to re-sign blocks or chunks on behalf of proxy in the course of user revocation.
- So here this paper suggests a predicate encryption with partial public key to avoid the re-generating services from recovering the whole data.
- To the best of the facts, this system is the first to support scalable and effective public auditing in the Cloud Computing. In particular, this system attains batch auditing where numerous delegated auditing jobs from diverse users can be performed simultaneously by the TPA.

By implementing the above appliances the privacy of outsourced data is retained well and also they can overcome the security threats. So this mechanism can well improve the efficiency of user revocation and computation and communication resources of existing users can be simply saved.

C. Design Goals

To appropriately and efficiently check the integrity of data/file and keep the stored file/data available for cloud storage, this proposed auditing system should attain the following properties:

- *Public Auditability:* To permit TPA to validate the intactness of the data/file in the cloud on demand without announcing additional online burden to the data owner.
- *Storage Soundness:* To confirm that the cloud server can never pass the auditing technique except when it really manages the owner's data/file intact.
- *Privacy Preserving:* To confirm that neither the auditor nor the Re-signature services can develop users' data/file content from the auditing and reparation procedure.
- *Authenticator Regeneration:* The authenticator of the repaired blocks/chunks can be exactly regenerated in the nonappearance of the data owner.
- *Error Location:* To confirm that the incorrect server can be quickly indicated when data exploitation is identified.

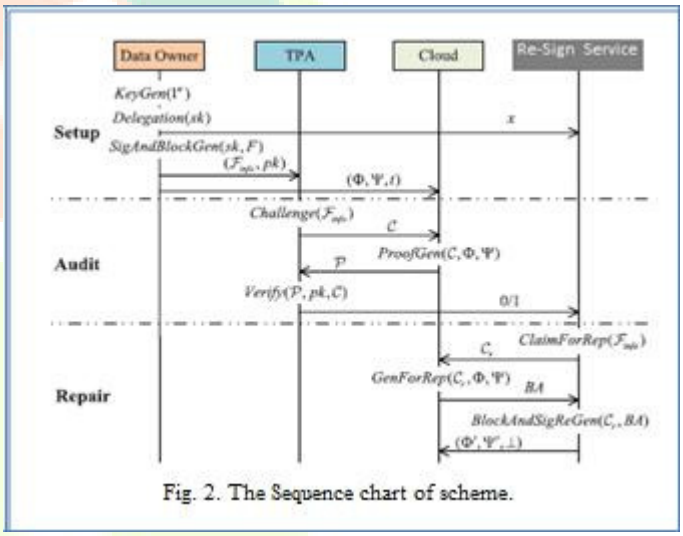

Fig. 2. The Sequence chart of scheme.

D. Definitions of Our Auditing Scheme

This auditing system consists of three procedures: Setup, Audit and Repair. Each technique holds polynomial-time algorithms as follows:

Setup: The data owner preserves this procedure to initialize the auditing system. KeyGen(1κ) → (pk, sk): This polynomial-time algorithm is run by the data owner to set its public and undisclosed parameters by captivating a security parameter κ as input.

Delegation(sk) → (x): This algorithm signifies the communication between the data owner and Re-signature services. The data owner provides partial secret key x to the proxy through a protected approach

SigAndBlockGen(sk, F) → (φ, ψ, t): This polynomial time algorithm is run by the data owner and receipts the undisclosed parameter sk and the novel file F as input, and then outputs a coded block/chunks set , an authenticator set and a file tag.

*Audit:*The cloud servers and TPA communicate with one another to take a arbitrary section on the blocks and verify the data intactness in this procedure.

*Challenge(F_{info})→ (C):*This algorithm is performed by the TPA with the information of the file F_{info} as input and a challenge C as output.

Proof Gen(C, φ, ψ)→(P): This algorithm is run by each cloud server with input challenge C, coded block set and authenticator set , then it outputs a proof P.

Verify(P, pk, C)→(0, 1): This algorithm is run by TPA immediately after a proof is received. Taking the proof P, public parameter pk and the corresponding challenge C as input, it outputs 1 if the verification passed and 0 otherwise.

Repair: In the absenteeism of the data owner, the Re-signature services converses with the cloud servers during this process to repair the incorrect server detected by the auditing procedures. Thus, an issue arises when trying to fix how to regenerate authenticators for the repaired blocks/chunks. A direct solution, which is accepted to make data owners handle the regeneration.

ClaimFor Rep(F_{info}) →(C_r): This algorithm is similar with the *Challenge()* algorithm in the Audit phase, but outputs a claim for repair C_r .

GenFor Rep(C_r, φ, ψ) →(BA): The cloud servers run this algorithm upon receiving the C_r and finally output the block and authenticators set BA with another two inputs φ, ψ.

BlockAndSigReGen(C_r, BA) →(φ', ψ', ⊥): The proxy implements this algorithm with the claim C_r and responses BA from each server as input, and outputs a new coded block set ψ' and authenticator set φ' if successful, outputting ⊥if otherwise.

III. The Proposed Scheme

In this section we start from an overview of our auditing scheme, and then describe the main scheme and discuss how to generalize our privacy-preserving public auditing scheme. Furthermore, we illustrate some optimized methods to improve its performance.

A. Overview

Although [7], [8] introduced private remote data checking schemes for regenerating-code-based cloud storage, there are still some other challenges for us to design a public auditable version. First, although a direct extension of the techniques in [2], [12], and [15] can realize public verifiability in the multi-servers setting by viewing each block as a set of segments and performing spot checking on them, such a straightforward method makes the data owner generate tags for all segments independently, thus resulting in high computational overhead. Considering that data owners commonly maintains limited computation and memory capacity, it is quite significant for us to reduce those overheads. Second, unlike cloud storage based on traditional erasure code or replication, a fixed file layout does not exist in the regenerating-code-based cloud storage. During the repair phase, it computes out new blocks, which are totally different from the faulty ones, with high probability.

Thus, a problem arises when trying to determine how to regenerate authenticators for the repaired blocks. A direct solution, which is adopted in [7], is to make data owners handle the regeneration. However, this solution is not practical because the data owners will not always remain online through the life-cycle of their data in the cloud, more typically, it becomes off-line even after data uploading. Another challenge is brought in by the proxy in our system model (see Section II-C). The following parts of this section shows our solution to the problems above. First, we construct a BLS-based [17] authenticator, which consists of two parts for each segment of coded blocks. Utilizing its homomorphic property and the linearity relation amongst the coded blocks, the data owner is able to generate those authenticators in a new method, which is more efficient compared to the straightforward approach. Our authenticator contains the information of encoding coefficients to avoid data pollution in the reparation with wrong coefficients. To reduce the bandwidth cost during the audit phase, we perform a batch verification over all α blocks at a certain server rather than checking the integrity of each block separately as [7] does. Moreover, to make our scheme secure against the replace attack and replay attack, information about the indexes of the server, blocks, and segments are all embedded into the authenticator. Besides, our primitive scheme can be easily improved to support privacy-preserving through the masking of the coding coefficients with a keyed PRF.

All the algebraic operations in our scheme work over the field GF (p) of modulo p, where p is a large prime (at least 80 bits).[4]

B. Construction of Our Auditing Scheme

Considering the regenerating-code-based cloud storage with parameters $(n, k, \ell, \alpha, \beta)$, we assume $\beta = 1$ for simplicity. Let G and G_T be multiplicative cyclic groups of the same large prime order p, and $e : G \times G \rightarrow G_T$ be a bilinear pairing map as introduced in the preliminaries. Let g be a generator of G and $H(\cdot) : \{0, 1\}^* \rightarrow G$ be a secure hash function that maps strings uniformly into group G. Table I list the primary notations and terminologies used in our scheme description.

Setup: The audit scheme related parameters are initialized in this procedure.

KeyGen(1^k) → (pk, sk): The data owner generates a random signing key pair *(spk, ssk)*, two random elements $x, y \in_R \leftarrow p$ and computes $pk_x \leftarrow g^x$, $pk_y \leftarrow g^y$. Then the secret parameter is $sk = (x, y, ssk)$ and the public parameter is $pk = (pk_x , pk_y, spk)$.

Delegation(sk) → (x): The data owner sends encrypted x to the cloud code regeneration service using the regeneration service's public key, then the regeneration service decrypts and stores it locally upon receiving.

IV. REGENERATING-CODE-BASED CLOUD

A. ADMIN MODULE

Admin, User and Third party registration

In this module, the Admin should register first. Then only he/she can be able to do user and group creation. For that admin needs to fill the information/details in the registration form. These information/details are maintained in a database.

User wants to access the file/data which is maintained in a cloud he/she should register their details first. These details are maintained in a Database. Admin will provide the access

to users and user should register their details and they can login with their user id and password and they can use the cloud space.

If a Third party auditor (TPA) wants to do some audit activity, they should register first. These details are maintained in a Database. Admin will provide the access to TPA and TPA should register their details and they can login with their user id and password and they can use the cloud space.

Group Creation

In this module, the admin is one who can create a new group and they could assign a group owner to someone from the user list.

B. DATA SHARING AND RESIGN MODULE

- *Block Verification Module*
- *Block Insertion Module*
- *Block Deletion module*

Shared data is divided into a number of blocks. A user in the group can modify the chunks in shared file/data by performing an insert, delete or update action on the chunks. To protect the integrity of shared file/data, each chunk in shared data is enclosed with a signature, which is computed by one of the users in the group/team. Specifically, when shared file/data is firstly created by the novel user in the cloud, all the signatures on shared file/data are computed by the novel user. After that, once a user changes a chunks, this user also needs to sign the modified chunks/block with his/her own private key. By sharing file/data among a group/team of users, different chunks/blocks may be signed by different users due to alterations from different users.

Block Verification Module

User can check that the uploaded file is modified by any other user or not.

Block Insertion Module

In the block insertion module user can insert the new block in already shared data which is initially created by different user.

Block Deletion module

In this user can delete the block which is initially created by different user.

C. USER REVOCATION AND BATCH AUDITING MODULE

When a user in the group/team leaves or behaves badly, the group needs to be revoked the particular user. Generally, as the creator of shared data, the original user acts as the group boss and he can capable to revoke users on behalf of the team/group. Once a user is revoked, the signatures added by this revoked user turn out to be invalid to the team/group, and the blocks/chunks that were previously signed by this revoked user should be re-signed by a surviving user's private key. So that the accuracy of the whole data/files can still be certified with the public keys of existing users only.

D. *CODE REGENERATION MODULE*

A public verifier is always can do audit the integrity of shared files/data without downloading the entire file/data from the cloud, even if some part of shared files/data has been re-signed by the cloud. In the absence of the data owner, the cloud re-generating services interacts with the cloud to repair the wrong server detected by the auditing process. The cloud Re-generating services regenerates the authenticators without downloading the revoked user's data by using predicate partial key of the user. So that the computation and communication resources of remaining users can be easily saved. This system achieves group auditing where multiple outsourced auditing jobs from various users can be done concurrently by the TPA.

In this section we start from an overview of our auditing scheme mechanism is further extended to support batch auditing. This system is based on maintaining the revoked user's information in, and then describe the main scheme and discuss how to generalize our privacy-preserving public auditing scheme.

To improve the efficiency of verifying group auditing responsibilities, the cloud servers and also data owner instead it being handled by third party auditor. The future work is how to prove data/file freshness (prove the cloud possesses the latest version of shared data) while still stabilizing identity privacy.

The system can be further enhanced by placing multiple servers so that the data can be retrieved even if a server fails and the performance can be improved.

V. RELATED WORK

The problem of remote data checking for integrity was first introduced in [26] and [27]. Then Ateniese *et al.* [2] and Juels and Kaliski [3] gave rise to the similar notions *provable data possession (PDP)* and *proof of retrievability (POR)*, respectively. Ateniese *et al.* [2] proposed a formal definition of the PDP model for ensuring possession of files on untrusted storage, introduced the concept of RSA-based homomorphic tags and suggested randomly sampling a few blocks of the file. In their subsequent work [28], they proposed a dynamic version of the prior PDP scheme based on MAC, which allows very basic block operations with limited functionality but block insertions. Simultaneously, Erway *et al.* [29] gave a formal framework for dynamic PDP and provided the first fully dynamic solution to support provable updates to stored data using rank-based authenticated skit lists and RSA trees. To improve the efficiency of dynamic PDP, Wang *et al.* [30] proposed a new method which uses merkle hash tree to support fully dynamic data.

To release the data owner from online burden for verification, [2] considered the public auditability in the PDP model for the first time. However, their variant protocol exposes the linear combination of samples and thus gives no data privacy guarantee. Then Wang *et al.* [14], [15] developed a random blind technique to address this problem in their BLS signature based public auditing scheme. Similarly, Worku *et al.* [31] introduced another privacy-preserving method, which is more efficient since it avoids involving a computationally intensive pairing operation for the sake of data blinding. Yang and Jia [9] presented a public PDP scheme, where the data privacy is provided through combining the cryptography method with the bilinearity property of bilinear pairing. [16]

utilized random mask to blind data blocks in error-correcting coded data for privacy-preserving auditing with TPA. Zhu *et al.* [10] proposed a formal framework for interactive provable data possession (IPDP) and a zero-knowledge IPDP solution for private clouds. Their ZK-IPDP protocol supports fully data dynamics, public verifiability and is also privacy-preserving against the verifiers.

VI. Conclusion

By means of suggesting privacy-preserving public auditing mechanism for re-generating code based data/file in the cloud, This scheme exploit cloud re-signature services to build a novel public authenticators, so that the revoked users information is also preserved by the cloud servers and also data owner instead of handled by third party auditor during corruptions. This scheme can well improve the efficiency of user revocation and computation and communication resources of current users can be easily saved.

REFERENCES

[1] M. Armbrust *et al.*, ―Above the clouds: A Berkeley view of cloud computing,‖ Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley,
CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009.

[2] G. Ateniese *et al.*, ―Provable data possession at untrusted stores,‖ in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2007, pp. 598–609.

[3] A. Juels and B. S. Kaliski, Jr., ―PORs: Proofs of retrievability for large files,‖ in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[4] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, ―MR-PDP: Multiple-replica provable data possession,‖ in *Proc. 28th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2008, pp. 411–420.

[5] K. D. Bowers, A. Juels, and A. Oprea, ―HAIL: A high-availability and integrity layer for cloud storage,‖ in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 187–198.

[6] J. He, Y. Zhang, G. Huang, Y. Shi, and J. Cao, ―Distributed data possession checking for securing multiple replicas in geographically dispersed clouds,‖ *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1345–1358, 2012.

[7] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, ―Remote data checking for network coding-based distributed storage systems,‖ in *Proc. ACM Workshop Cloud Comput. Secur. Workshop*, 2010, pp. 31–42.

[8] H. C. H. Chen and P. P. C. Lee, ―Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation,‖ *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 407–416, Feb. 2014.

[9] K. Yang and X. Jia, ―An efficient and secure dynamic auditing protocol for data storage in cloud computing,‖ *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[10] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, ―Cooperative provable data possession for integrity verification in multicloud storage,‖ *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, Dec. 2012.

[11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, ―A survey on network codes for distributed storage,‖ *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, Mar. 2011.

[12] H. Shacham and B. Waters, ―Compact proofs of retrievability,‖ in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2008, pp. 90–107.

[13] Y. Hu, H. C. H. Chen, P. P. C. Lee, and Y. Tang, ―NCCloud: Applying network coding for the storage repair in a cloud-of-clouds,‖ in *Proc. USENIX FAST*, 2012, p. 21.

[14] C. Wang, Q. Wang, K. Ren, and W. Lou, ―Privacy-preserving public auditing for data storage security in cloud computing,‖ in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[15] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, ―Privacy-preserving public auditing for secure cloud storage,‖ *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[16] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, ―Toward secure and dependable storage services in cloud computing,‖ *IEEE Trans. Service Comput.*, vol. 5, no. 2, pp. 220–232, Apr./Jun. 2012.

[17] D. Boneh, B. Lynn, and H. Shacham, ―Short signatures from the Weil pairing,‖ *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.

[18] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, ―Network coding for distributed storage systems,‖ *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[19] T. Ho *et al.*, ―A random linear network coding approach to multicast,‖ *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[20] D. Boneh, D. Freeman, J. Katz, and B. Waters, ―Signing a linear subspace: Signature schemes for network coding,‖ in *Public Key Cryptography*. Berlin, Germany: Springer-Verlag, 2009, pp. 68–87.

[21] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, ―Dynamic provable data possession,‖ in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213–222.

[22] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, ―Enabling public verifiability and data dynamics for storage security in cloud computing,‖

[23] S. G. Worku, C. Xu, J. Zhao, and X. He, ―Secure and efficient privacy preserving public auditing scheme for cloud storage,‖ *Comput. Elect. Eng.*, vol. 40, no. 5, pp. 1703–1713, 2013.

IJARM
Your all-MATE