International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016 [ICSSCCET 2016]

ISBN	978-81-929866-6-1	**VOL**	02
Website	icssccet.org	**eMail**	icssccet@asdf.res.in
Received	25 – February – 2016	**Accepted**	10 - March – 2016
Article ID	ICSSCCET049	**eAID**	ICSSCCET.2016.049

An Efficient Cloud Storage Batch Auditing without Key Exposure Resistance using Public Verifier

T Yawanikha[1], R Meyanand[2], T Dhivya[3], B Monica[4], R R Minty[5], S R Subashini[6]

[1]Assistant Professor, Department of Information Technology, Karpagam Institute of Technology, Coimbatore

[2]Assistant Professor, [3,4,5,6] UG Scholar, Department of Information Technology, Selvam College of Technology, Namakkal

Abstract: *Cloud Computing is a paradigm which provides services over network. In Cloud computing, data is usually audited by an external auditor in public cloud. Naive solution and slightly better solution are the two methods that authenticates client with key. Using build in key, previously stored data can be verified but the problems due to key exposure are not addressed. In proposed system, Cloud storage auditing is done without key exposure. Public Aware data sharing Public Verifier (PAPV) algorithm is employed to private users before exposing data. In addition to this,Batch system is allocated to organize time period independently which is an abstraction to the tenant' s secret key in public cloud. Further, block verification is developed to support data security. By implementing this, the high end secure data can be obtained to any user who stores and manipulates data in cloud. Hereby, the proof analysis developed reveal that our proposed is efficient to tenants in public cloud.*

Index terms: *Data Storage, Batch auditing, Public Verifier, Data Cloud Server.*

1. INTRODUCTION

Cloud Computing is a paradigm where large pool of systems are connected in private or public networks to provide dynamically scalable infrastructure for application, data and file storage. The outsourced storage in clouds has become a new profit growth point by providing a comparable low cost, scalable, location independent platform for managing client's data. The cloud storage relieves the burden for storage management and maintenance. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. Security audit is an important solution enabling trace back and analysis of any activities including data access, security breaches, application activities and so on.

Third party auditor is an accepted method for establishment trust between two parties with potentially different incentives [2]. Auditors assess and expose risk, enabling customers to choose rationally between competing services. We believe auditing is necessary not only for traditional business but also for online services. One way to rely on a trusted third party auditor, who has sufficient access to the provider's environment. An auditor understands the service level agreement (SLA) between a customer and a provider and quantifies the extent to which the provider might not meet the SLA. We identify auditing by two approaches external and internal auditing. Considering the role of verifier in the model, all the schemes presented before fall into two categories: private and public auditability.[2]Although schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone not just the client to challenge The cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the information of the service performance to the independent third party auditor without devotion of their computational resources. Another major concern among previous designs is that of supporting dynamic data operation for cloud data storage applications. In Cloud Computing, the remotely stored electronic data might not only be accessed but also updated by the clients, e.g.,

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

through block modification, deletion and insertion, etc. Unfortunately, the state-of the -art in the context of remote data storage mainly focus on static data files and the importance of this Dynamic data update has received limited attention so far [7], [1]. Moreover, as will be shown later, the direct extension of the current provable data possession (PDP) [2] or proof of retrievability (POR) [4] with schemes to support data dynamics may lead to security loopholes. Although there are many difficulties faced by researchers, it is well believed that supporting dynamic data operation can be of vital importance to the practical application of storage outsourcing services. In view of the key role of public auditability and data dynamics for cloud data storage, we propose an efficient construction for the seamless integration of these components in the protocol design. We deal with client's secret key exposure which is a major concern to the protocol in cloud storage. In previous work, protocols designed didn't consider the problems faced due to the exposure of key in public cloud. In this paper, we focus on how to reduce the problems such as allowing duplicate data, privacy problems, computational time and energy consumption due to audit using remote servers. Previous process involves retrieving whole data or the data that is known to verify but in this design we create a private security for every user by creating groups in public cloud. Secondly, adopt the standards of outsourcing the data without the knowledge and exposure of key in either public or private cloud. Here, the auditing [10] is ensured by blocked verification in public cloud.

The blocked audit is done by batch auditing where each user in the group specified with a key which is private to the original user. Our experimental results not only validate the effectiveness of our approaches, but also show that our system does not create any significant computation cost and require less extra storage for integrity verification. The aggregation and algebraic properties of the authenticator further benefit our design for the batch auditing purpose. Specifically, our contribution can be summarized as the following two aspects:

- We motivate the public auditing system of data storage security in Cloud Computing and provide a public aware for privacy data sharing using public verifier (PAPV),i.e., our scheme enables an auditor in cloud to audit user's outsourced data without learning the data content.
- To the best of our knowledge, our scheme is the first to support scalable and efficient public auditing in the Cloud Computing.

Specifically, our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by Public verifier.

2. Background and Related Work

In previous researches, to implement public auditability, the notions of proof of retrievability (POR) [2] and PDP [5] have been proposed by some researchers. These approaches were based on a probabilistic proof technique for a storage provider to prove that clients' data remain intact. For ease of use, some POR/PDP schemes work on a publicly verifiable way, so that anyone can use the verification protocol to prove the availability of the stored data. Hence, they help accommodate the requirements from public auditability. POR/ PDP schemes evolved around an untrusted storage offer a publicly accessible remote interface to check the tremendous amount of data.

In terms of key exposure implemented in earlier study, we firstly show two basic solutions for the key-exposure problem of cloud storage auditing before we give our core protocol. [9]The first is a naive solution, which in fact cannot fundamentally solve this problem. The second is a slightly better solution, which can solve this problem but has a large overhead. They are both impractical when applied in realistic settings. And then we give our core protocol that is much more efficient than both of the basic solutions.

2.1 Naive Solution

In this solution, the client still uses the traditional key revocation method. Once the client knows his secret key for cloud storage auditing is exposed, he will revoke this secret key and the corresponding public key. Meanwhile, he generates one new pair of secret key and public key, and publishes the new public key by the certificate update.

The authenticators of the data previously stored in cloud, however, all need to be updated because the old secret key is no longer secure. [3]Thus, the client needs to download all his previously stored data from the cloud, produce new authenticators for them using the new secret key and then upload these new authenticators to the cloud. Obviously, it is a complex procedure and consumes a lot of time and resource Furthermore, because the cloud has known the original secret key for cloud storage auditing, it may have already changed the data blocks and the corresponding authenticators. It would become very difficult for the client to even ensure the correctness of downloaded data and the authenticators from the cloud. Therefore, simply renewing secret key and public key cannot fundamentally solve this problem in full.

2.2 Slightly Better Solution

The client initially generates a series of public keys and secret keys: (PK1, SK1), (PK2, SK2). . . (PKT, SKT). Let the fixed public key be (PK1. PKT) and the secret key in time period j be (SK j, SKT). If the client uploads files to the cloud in time period j, the client

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

uses SK j to compute authenticators [7] [3] for these files. Then the client uploads files and authenticators to the cloud. When auditing these files, the client uses PKj to verify whether the authenticators for these files are indeed generated through SK j. When the time period changes from j to j + 1, the client deletes SK j from his storage. Then the new secret key is than the naive solution. These solutions represent the key to be generated with them hence ensures confidentiality of the audited data.

To fully ensure the data integrity and save the cloud users computation resources as well as online burden, it is of critical importance to enable public auditing service for cloud data storage, so that users may resort to an independent third party auditor (TPA) to audit the outsourced data when needed. The figure 2.2 represents the verification of data using third party auditor (TPA) with a data server and name server. The name server allows to find the identity of the data in the cloud when data is exposed for auditing. Finally, the queries are handled by the authorized applications of the specified data owners.

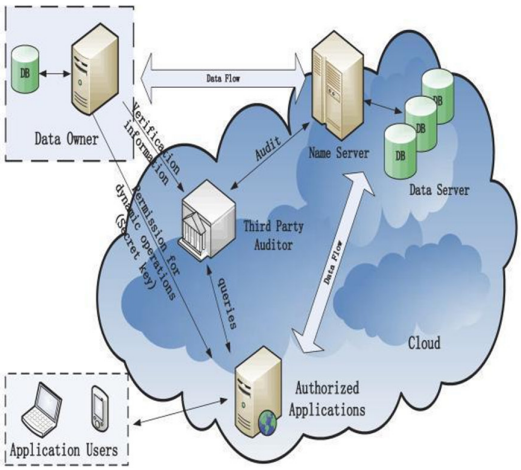

Fig 2.2: Architecture of cloud verification

The TPA, who has expertise and capabilities that users do not, can periodically check the integrity of all the data stored in the cloud on behalf of the users, which provides a much more easier and affordable way for the users to ensure their storage correctness in the cloud. Moreover, in addition to help users to evaluate the risk of their subscribed cloud data services, the audit result from TPA would also be beneficial for the cloud service providers to improve their cloud based service platform, and even serve for independent arbitration purposes [9]. In a word, enabling public auditing services will play an important role for this nascent cloud economy to become fully established; where users will need ways to assess risk and gain trust in the cloud.

3. Problem Statement

3.1 System Model and Definition

A cloud data storage service involves three different entities, as illustrated previous papers the cloud user (U), who has large amount of data files to be stored in the cloud; the cloud server (CS), which is managed by the cloud service provider (CSP) to provide data storage service and has significant storage space computational resource(we will not differentiate CS and CSP hereafter); the third party auditor (TPA), who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service reliability on behalf of the user upon request. Users rely on the CS for cloud data storage and maintenance. They may also dynamically interact with the CS to access and update their stored data for various application purposes. To save the computation resource as well as the online burden, cloud users may resort to [4][5] TPA for ensuring the storage integrity of their outsourced data, while hoping to keep their data private from TPA. Here Public verifier (PAVA) is used instead of TPA to maintain the data within each cloud environment. Representative network architecture for cloud data storage is illustrated. Three different network entities can be identified as follows:

- Client: an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations;
- Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the client's data.
- Third Party Auditor (TPA): an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

In the cloud paradigm, by putting the large data files on the remote servers, the clients can be relieved of the burden of storage and computation. As clients no longer possess their data locally, it is of critical importance for the clients to ensure that their data are being correctly stored and maintained. That is clients should be equipped with certain security means so that they can periodically verify the correctness of the remote data even without the existence of local copies.

In case that client does not necessarily have the time, feasibility or resources to monitor their data, they can delegate the monitoring task to a trusted TPA. In this paper, we only consider verification schemes with public auditability: any TPA in [5] possession of the public key can act as a verifier. We assume that TPA is unbiased while the server is untrusted. For application purposes, the clients may interact with the cloud servers via CSP to access or retrieve their pre-stored data. More importantly, in practical scenarios, the client may frequently perform block-level operations on the data files.

The most general forms of these operations we consider in this paper are modification, insertion, and deletion. These orthogonal problems are based on the verification of public key and private key using frequency generated by cloud privacy.

4. Design Goals

To enable privacy-preserving public auditing for cloud data storage under the aforementioned model, our protocol design should achieve the following security.

- **Public auditability:** to allow TPA to verify the correctness of the cloud data on demands retrieving a copy of the whole data or introducing additional online burden to the cloud users.
- **Storage correctness:** to ensure that there exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.
- **Privacy-preserving:** to ensure that the TPA cannot derive users' data content from the information collected during the auditing process.
- **Batch auditing:** to enable TPA with secure and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different users simultaneously.
- **Lightweight:** to allow TPA to perform auditing with minimum communication and computation overhead.

Public auditability for storage correctness assurance: to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand; Dynamic data operation support to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the [9][8] seamless integration of public auditability and dynamic data operation support. Block less verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern. To authorize the CS to respond to the audit delegated to TPA's, the user can sign a certificate granting audit rights to the TPA's public key, and all audits.

5. Architecture and Techniques

Audit system architecture for outsourced data in clouds is described where we consider that a data storage service involves four entities:

DO, who has a large amount of data to be stored in the cloud; CSP, who provides data storage service and has enough storage space and computation resources; TPA, who has capabilities to manage or monitor the outsourced data under the delegation of DO; and authorized applications (AAs), who have the right to access and manipulate the stored data. Finally, application users can enjoy various through the following audit functions: TPA should be able to make regular checks on the integrity and availability of the delegated data at appropriate assume the TPA is reliable and independent intervals.

Tag generation: The client (DO) uses a secret key to preprocess a file, which consists of a collection of n blocks, generates a set of public verification parameters (PVPs) and IHT that are stored in TPA, transmits the file and some verification tags to CSP,[6]and may delete its local copy. Periodic sampling audit. By using an interactive proof protocol of retrievability, TPA (or other applications) issues "random sampling" challenge to audit the integrity and availability of the outsourced data in terms of verification information (involving PVP and IHT) stored in TPA and Audit for dynamic operations.

An AA, who holds a DO's secret key sk, can manipulate the outsourced data and update the associated IHT stored in TPA. The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the AAs and forge the valid audit records In general, the AAs should be cloud application services inside clouds for various application purposes, but they must be specifically modifications for data will be detected in audit processes or verification processes.

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

Based on this kind of strong authorization-verification mechanism, we assume neither CSP is trusted to guarantee the security of stored data, nor a DO has the capability to collect CSP's faults after errors have been found.

The ultimate goal of this audit infrastructure is to enhance the credibility of CSSs, but not to increase DO's burden. The fig 5.1 represents the flow of security messages through the servers in cloud with the authorized user and cloud service provider where user outsources their files for auditing.

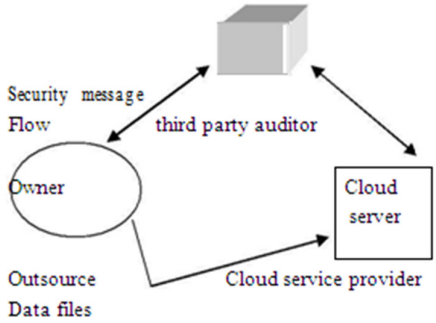

Fig 5.1: Data cloud server

The cloud service provider is no different from cloud server. Owner is responsible for the message encryption and decryption. Therefore, TPA should be constructed in clouds and maintained by a CSP. To ensure the trust and security, TPA must be secure enough to resist them which should be strictly controlled to prevent unauthorized accesses even for internal members in clouds. A more practical trusted third party (TTP). This mechanism not only improves the performance of an audit service, but also provides the DO with a maximum access transparency. This means that DOs are entitled to utilize the audit service without additional costs.

6. The Proposed Schemes

Public auditing scheme provides a complete outsourcing solution of data– not only the data itself, but also its integrity checking. We start from an overview of our public auditing system and discuss two straightforward schemes and their demerits. Then we present our main scheme and show how to extent our main scheme to support batch auditing for the TPA upon delegations from multiple users. Finally, we discuss how to generalize our privacy-preserving public auditing scheme and its support of data dynamics. There are three main login account cloud login, public verifier login and user login. The following are the proposed models based on public auditing.

6.1 Group member registration and login
6.2 Batch level sign based key generation
6.3 Upload files to cloud server
6.4 Download files from cloud server
6.5 Public auditing with user revocation in Public Verifier.

6.1 Group Member Registration and Login

In this module the first User entered his username, password, and chooses any one group id then register with Data Cloud Server. This user added in this particular group. Then he entered this username, password and choose his group id to login.

6.2 Batch Level Sign Based Key Generation

In Key Generation module, every user in the group generates his/her public key and private key. User Generates a random p, and outputs public key and private key. Without loss of generality, we assume user u1 is the original user, who is the creator of shared data. The original user also creates a user list (UL), which contains ids of all the users in the group. The user list is public and signed by the original user.

6.3 Upload Files to Cloud Server

In this module the user wants to upload a file. So he split the files into many blocks. Next he encrypt each blocks with his public key. Then he generate signature of each blocks for authentication purpose. Then he upload each block cipher text with signature, block id and signer id. These metadata and Key Details are stored in Public Verifier for public auditing. If any key generated is invalid then user revocation is done by original user.

6.4 Download Files from Cloud Server

In this module the next user or group member wants to download a file. So he gives the filename and get the secret key. Then he entered this secret key. If this secret key is valid then the user able to decrypt this downloaded file. Else heentered wrong secret key

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

then he blocked by Public Verifier. If this secret key is valid then decrypt each block and verify the signature. If both signatures are equal then combine all blocks then get the original file.

6.5 Public Auditing with User Revocation in Public Verifier

In this module, the User who entered the wrong secret key then he blocked by the public verifier. Next he added public verifier revoked user list. Then he wants to tries to download any file, the Data Cloud Server replies his blocked information. Then he wants to unrevocation, so he ask the public verifier. Finally the public verifier unrevoked this user. Next he able to download any file with its corresponding secret key.

7. Basic Solution

Assume the outsourced data file F consists of a finite ordered set of blocks m1, m2, . .. , mn. One straightforward way to ensure the data integrity is to pre-compute MACs for the entire data file. Specifically, before data outsourcing, the data owner pre-computes MACs of F with a set of secret keys and stores them locally. During the auditing process, the data owner each time reveals a secret key to the cloud server and asks for a fresh keyed MAC for verification.

This approach provides deterministic data integrity assurance Straight forwardly as the verification covers all the data blocks. However, the number of verifications [5] allowed to be performed in this solution is limited by the number of secret keys. Once the keys are exhausted, the data owner has to retrieve the entire file of F from the server in order to compute new MACs, which is usually impractical due to the huge communication overhead.

Moreover, public auditability is not supported as the private keys are required for verification. Another basic solution is to use signatures instead of MACs to obtain public auditability [6].The data owner precomputes the signature of each block mi (i ∈ [1, n]) and sends both F and the signatures to the cloud server for storage.

To verify the correctness of F, the data owner can adopt a spot-checking approach, i.e., requesting a number of randomly selected blocks and their corresponding signatures to be returned. This basic solution can provide probabilistic assurance of the data correctness and support public auditability. Fig 7.1 shows verifying public data using public verifier algorithm. Each user needs to verify with a single verifier in cloud. Auditing channels and auditing proof are generated in cloud servers. However, it also severely suffers from the fact that a considerable number of original data blocks should be retrieved to ensure a reasonable detection probability, which again could result in a large communication overhead and greatly affects system efficiency. Notice that the above solutions can only support the case of static data and none of them can deal with dynamic data updates. Our protocol execution is as follows:

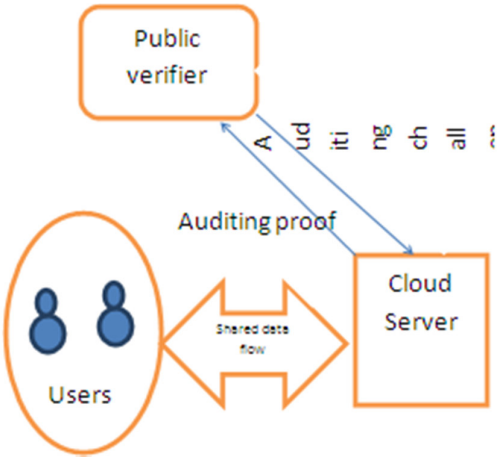

Fig 7.1: Public Verifier in cloud

8. Construction and Verification

To effectively support public auditability without having to retrieve the data blocks themselves, we resort to the homomorphic authenticator technique [2][4]. Homomorphic authenticators are unforgeable metadata generated from individual data blocks, which can be securely aggregated in such a way to assure a verifier that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. In our design, we propose to use PKC based homomorphic authenticator (e.g., BLS signature [4] or RSA signature based authenticator [2]) to equip the verification protocol with public auditability.

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

In the following description, we present the BLS-based scheme to illustrate our design with data dynamics support. As will be shown, the schemes designed under BLS scheme Construction can also be implemented in RSA construction. In the discussion of section III-D, we show that direct extensions of previous work [2], [4] have security problems. And we believe that protocol design for supporting dynamic data operation is a major challenging task for cloud storage systems. Now we start to present the main idea behind our scheme. We assume that file F (potentially encoded using Reed-Solomon codes [15]) is divided into n blocks m1,m2, . . . ,mn 1, where mi ∈Zp and p is a large prime. Let e : G×G → GT be a bilinear map, with a hash function H : {0, 1}∗ → G, viewed as a random oracle [19]. Let g be the generator of G. h is a cryptographic hash function. The procedure of Setup: The client's public key and private key are generated by invoking KeyGen(•). By running SigGen(•), the data file F is pre-processed, and the homomorphic authenticators together with metadata are produced.

KeyGen(1k). The client generates a random signing key. SigGen(sk, F). Given F = (m1,m2 . . . ,mn), the client chooses a random Element u ← G. Let t=name || n|| u || SSigssk(name||n||u) be the file tag for F. Then the client computes signature σi for each block mi (i = 1, 2, . . . , n) as σi ← (H(mi) • umi)α. Denote the set of signatures by _ = {σi}, 1 ≤ i ≤ n. The client then generates a root R based on the construction.

8.1 Default Integrity Verification

The client or TPA can verify the integrity of the outsourced data by challenging the server. Before challenging, the TPA first usespk to verify the signature on t. If the verification fails, reject by emitting FALSE; otherwise, recover u. To generate the message "chal", [4][10]the TPA (verifier) picks a random celement subset I = {s1, s2, . . . , sc} of set [1, n], where we assume s1 ≤ • • • ≤ sc. For each i ∈ I the TPA chooses a random element νi ← B ⊆Zp. The message "chal" specifies the positions of the blocks to be checked in this challenge phase.

The verifier sends the chal {(i, νi)}s1≤i≤sc to the prover (server). GenProof(F,_, chal). Upon receiving the challenge chal = {(i, νi)}s1≤i≤sc , the server computes μ = sc X i=s1 νimi∈Zp and σ = sc Yi=s1 σνi i ∈ G, where both the data blocks and the corresponding signature blocks are aggregated into a single block, respectively. In addition, the prover will also provide the verifier with a small amount of auxiliary information {i}s1≤i≤sc , which are the node siblings on the path from the leaves {h(H(mi))} s1≤i≤sc to the root R of the MHT. The prover responds the verifier with proof P = {μ, σ, {H(mi),i} s1≤i≤sc , scsigsk (H(R))}. Verify Proof(pk, chal, P). Upon receiving the responses from the prover, the verifier generates root R using {H(mi),i}s1≤i≤sc and authenticates it by checking e(sigsk(H(R)), g)?= e(H(R), gα). If the authentication fails, the verifier rejects by emitting FALSE. Otherwise, the verifier checks e(σ, g) ?= e(sc Y i=s1 H(mi)νi • uμ, v). If so, output TRUE; otherwise FALSE.

8.2 Dynamic Data Operation with Integrity Assurance

Now we show how our scheme can explicitly and efficiently handle fully dynamic data operations including data modification (M), data insertion (I) and data deletion (D) for cloud data storage. Note that in the following descriptions, we assume that the file F and the signature _ have already been generated and properly stored at server. The root metadata R has been signed by the client and stored at the cloud server, so that anyone who has the client's public key can challenge the correctness of data storage.

8.3 Data Modification

We start from data modification, which is one of the most frequently used operations in cloud data storage. A basic data modification operation new ones. Suppose the client wants to modify thei -th block mi to m′ i.

The protocol procedures are described as At start, based on the new block m′ i, the client generates the corresponding signature σ′i = (H (m′ i) • um′ i)α. Then, he constructs an update request message "update = (M, i,m′ i, σ′i)" and sends to the server, where M denotes the modification operation. Upon receiving the request, the server runs ExecUpdate(F,_, update).Specifically, the server replaces as followshere

- replaces the block mi with m′ i and output.
- replaces the σi with σ′i and outputs _′;
- replaces H(mi) with H(m′)

8.4Designs for Distributed Data Storage Security

To further enhance the availability of the data storage security, individual user's data can be redundantly stored in multiple physical locations. That is, besides being exploited at individual servers, data redundancy can also be employed across multiple servers to tolerate faults or server crashes as user's data grows in size and importance. It is well known that erasure-correcting code can be used to tolerate multiple failures in distributed storage systems.

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

In cloud data storage, we can rely on this technique to disperse the data file F redundantly across a setof n = m+k distributed servers. A [m+k, k]-Reed-Solomon code is used to create k redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the m+k data and parity vectors.

Using each of the m + k vectors on a different server, the original data file can survive the failure of any k of the m+k servers without any data loss. Such a distributed cryptographic system allows a set of servers to prove to a client that a stored file is intact and retrievable. The results are evaluated using the above methods using keys (k).

9. Support for Batch Auditing

With the establishment of privacy-preserving public auditing, the TPA may concurrently handle multiple auditing upon different users' delegation. The individual auditing of these tasks for the TPA can be tedious and very inefficient. Given Kkn auditing delegations on K distinct data files from K different users, it is more advantageous for the TPA to batch these multiple tasks together and audit at one time. Keeping this natural demand in mind, we slightly modify the protocol in a single user case, and achieves the aggregation of K verification equations (for K auditing tasks) into a single one.

As a result, a secure batch auditing protocol for simultaneous auditing of multiple tasks is obtained. The details are described as follows.

Setup Phase: Basically, the users just perform Setup independently. Suppose there are K users in the system, and each user k has a data file Fk = (mk,1, . . . ,mk,n) to be outsourced to the cloud server, where k ∈ {1, . . . ,K}. For simplicity, we assume each file Fk has the same number of n blocks. For a particular user k, denote his/her secret key as (xk, sskk), and the corresponding public parameter as (spkk, vk, g, uk, e(uk, vk)) where vk = gxk. Similar to the single user case, each user k has already randomly chosen a different (with overwhelming probability) name namek∈Zp for his / her file Fk, and has correctly generated the corresponding file tag tk = namek||SSigsskk (namek). Then, each user k runs SigGen and computes _k,i for block mk,i: _k,i ← (H(namek||i) . umk,I k). (i∈ {1, . . . , n}), where Wk,i = namek||i. Finally, each user k sends file Fk set of authenticators _k, and tag tk to the server and deletes them from the block of local storage.

9.1 Supports for Data Dynamics

In Cloud Computing, outsourced data might not only be accessed but also updated frequently by users for various application purposes [10], [12][2]. Hence, supporting data dynamics for privacy preserving public auditing is also of paramount importance. Now we show how to build upon the existing work [10] and adapt our main scheme to support data dynamics, including block level operations of modification, deletion and insertion. In [10], data dynamics support is achieved by replacing the index information i with mi in the computation of block authenticators and using the classic data structure – Merkle hash tree (MHT) [7]for the underlying block sequence enforcement. As a result, the authenticator for each block is changed to _i =(H(mi) • umi)x. We can adopt this technique inour design to achieve privacy-preserving public risk auditing with support of data dynamics. Specifically, in the Setup phase, the user has to generate and send the tree root TRMHT to TPA as additional metadata, where the leaf nodes of MHT are values of H(mi).

In the Audit phase, besides {μ, _,R}, the server's response should also include {H(mi)}i∈ I and their corresponding auxiliary authentication information aux in the MHT. Upon receiving the response, TPA should first use TRMHT and aux to authenticate {H(mi)}i∈ I computed by the server. Once {H(mi)}i∈ I are authenticated, TPA can then perform the auditing on {μ, _,R, {H(mi)}i∈ I} via Equation 1, where Qs1≤i≤sc H(Wi)_i is now replaced by Qs1≤i≤sc H(mi)_i . Data privacy is still preserved due to the random mask.

The details of handling dynamic operations are similar to [10] and thus omitted. using valid _ from the response. Specifically, the TPA can always guess whether the stored data contains certain message ˜m, by checking e(_, g) ? = e((Qsc i= s1 H(Wi)_i) u˜μ′ , v), where ˜μ′ is constructed from random coefficients chosen by the TPA in the challenge and the guessed message ˜m.Thus, our main scheme is not semantically secure yet. However, settings. Therefore, the TPA has to test basically all or-nothing guess launched by TPA can be Data privacy is still preserved due to the random mask. The details of handling dynamic operations are similar to [10] and thus omitted. using valid _ from the response. Specifically, the TPA negligible. Nonetheless, for completeness, we will give a provably zero-knowledge based public auditing scheme, which further eliminates the possibilities of above offline guessing attack. The details and corresponding proofs can be found.

Specifically, in the Setup phase, the user has to generate and send the tree root TRMHT to TPA as additional metadata.

10. Performance and Evaluation

Audit activities would increase the computation and communication overheads of the audit service. However, the less frequent activities may not detect anomalies in a timely manner. Hence, the scheduling of audit activities is significant for improving the quality of audit services.

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.

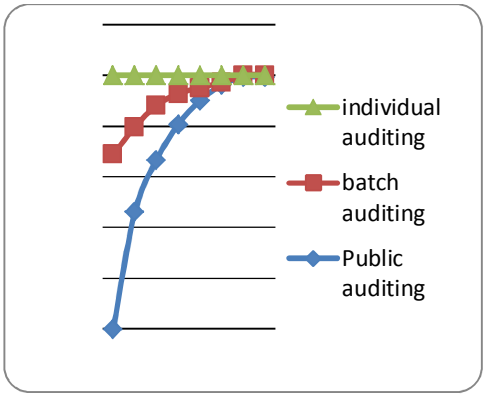

Fig 10.1: Evaluation of Batch auditing

The graph above fig 10.1 shows the variation between batch audition and public verifier auditing. Individual auditing do not vary between users who switch from one cloud to other.

To detect anomalies in a low overhead and timely manner we attempt to optimize and timely manner we attempt to optimize the audit performance from two aspects: performance evaluation of probabilistic queries and scheduling maintain a tradeoff between overhead and accuracy which helps us to improve the performance of audit systems.

11. Conclusion and Future Work

In this paper, we study on how to deal with the client's key without exposing into the cloud. The auditing performed by public verifier not only audits the data but also verifies the integrity of the data in cloud. The concept of user revocation allows to revoke the invalid key registered. We formalize the definition and the security model of auditing protocol without key-exposure resilience, and then propose and verify the first practical solution. Further the duplicated files are prohibited but do not address the issues due to creation of such files. In future we need to identify the solution for providing privacy to data that is not verified in public cloud.

References

1. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peter-son, and D. Song, "Provable data possession at untrusted stores,"inProc. of CCS'07New York, NY, USA: ACM, 2007, pp. 598–609.
2. M. Bellare and S. Miner, "A forward-secure digital signature scheme," in Advances in Cryptology—CRYPTO. Berlin, Germany: Springer-Verlag, 1999, pp.431–448.
3. Cong Wang, Sherman S.M. Chow, Qian Wang, KuiRen, and Wenjing Lou, Privacy Preserving Public Auditing for Secure Cloud Storage, IEEE Transactions on Computers (TC), 10, 451.
4. 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.
5. T. Stewart. (Aug. 2012). Security Policy and Key Management: Centrally ManageEncryption Key. [Online]. Available: http://www.slideshare.net/Tinastewart/security-policy-andenterprisekey
6. M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in Proc. 11th USENIX Workshop Hot Topics Oper. Syst. , 2007, pp. 1–6.
7. C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreserving public auditing for secure cloud storage," IEEE Trans.Comput., vol. 62, no. 2, pp. 362–375, Feb. 2013.
8. Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing,"IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847–859, May 2011.
9. H. Wang, "Proxy provable data possession in public clouds," IEEE Trans. Services Comput., vol. 6, no. 4, pp. 551–559, Oct./Dec. 2013.
10. B. Wang, B. Li, and H. Li, "Public auditing for shared data with efficient user revocation in the cloud," in Proc. IEEE INFOCOM, Apr. 2013, pp. 2904–2912.
11. K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. Parallel Distrib. Syst.,
12. Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds'IEEE TRANS service Comput., vol. 6, no. 2, pp. 227–238, Apr./Jun. 2013.

Cite this article as: T Yawanikha, R Meyanand, T Dhivya, B Monica, R R Minty, S R Subashini. "An Efficient Cloud Storage Batch Auditing without KEY Exposure Resistance using Public Verifier". *International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016*: 258-266. Print.