A Futuristic Work on Distributed Storage and Multiple Audition in Cloud Computing

Parkavi[1], Krishnapriya[2]

[1]M.Phil Scholar, Department of Computer Science,
Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamilnadu, India

[2]Head of the Department, Department of Computer Science
Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamilnadu, India

Abstract: ***A flexible distributed storage enhances integrity auditing mechanism, utilizing the homomorphism token. The distributed erasure-coded data also enabled. The proposed design is concentrated mainly to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization. The fast data error localization is nothing but the identification of misbehaving server. However the cloud data are dynamic in nature, the proposed design further supports secure and efficient dynamic operations on outsourced data. The operation includes block modification, deletion, and append. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure. It is nothing but the malicious data modification attack, and even server colluding attacks.***

Keywords: Auditing mechanism, Erasure coded data, Benzantine failure.

1. Introduction

A computer network or data network is a telecommunications network that allows computers to exchange data. In computer networks, networked computing devices (network nodes) pass data to each other along data connections. The connections (network links) between nodes are established using either cable media or wireless media. Network devices that originate, route and terminate the data are called network nodes. Nodes can include hosts such as servers and personal computers, as well as networking hardware. Two devices are said to be networked when a device is able to exchange information with another device.

1.1 Cloud Computing

Cloud computing is a new computational paradigm that offers an innovative business model for organization*s* to adopt it without upfront investments. Cloud computing is clearly one of today's most enticing technological breakthrough due to its cost-efficiency and flexibility. Though cloud computing enables the movement of application, software and data to a large data center where the data management and associated services may not be fully trustworthy thus raising some unanswered questions about data security.

As the Internet has increased in speed and bandwidth, remote storage of data over the network has become feasible. Peer-to-peer (P2P) storage systems, especially those based on the so-called Distributed Object Location and Retrieval (DOLR) systems [12] such as Oceanstore [10] are an important class of such systems. Systems like these face a number of challenges such as data privacy, protection of the data against alteration, data loss due to node unavailability and the *free rider* problem.

The techniques based on algebraic signatures that allow a "data origination site" to verify that a remote site is storing data correctly, or whether a number of sites that collectively store a collection of objects are doing so correctly. The scheme or techniques does not need the original data for its check, but only two small messages need to be exchanged for each check. Both of these properties should be attractive to designers of remote storage schemes.

As peer-to-peer technology has matured, a number of systems such as Oceanstore [10], Intermemory [14], Ivy [8], PAST [9], Starfish [13], FarSite [15] have beenbuilt to utilize remote data storage. To protect against failure, this data is stored redundantly using either pure replication or *m*/*n* erasure coding. Similarly, Lillibridge, *et al.* [11] propose a scheme where participants mutually store each other's backup data. All these schemes store data on sites that cannot be trusted. In addition to peer unavailability, they must face the problem of *free riders*. Free riders only pretend to store other's data and thus enjoy the benefits of remote storage of their data without incurring any costs of their own.

1.2 Cloud Storage

Cloud storage is a model of networked enterprise storage where data is stored not only in the user's computer, but in virtualized pools of storage which are generally hosted by third parties, too. Hosting companies operate large data centers, and people who require their data to be hosted either buy or lease storage capacity from them.

Cloud storage services may be accessed through a web service application programming interface (API), a cloud storage gateway or through a web-based user interface. An online data-backup service is a bounding for consumers and enterprises alike. Amazon Simple Storage Service (S3) [1], for example, offers an abstracted online-storage interface, allowing programmers to access data objects through web-service calls, with fees metered in gigabyte-months and data-transfer amounts. Researchers have investigated alternative service models, such as peer-to-peer data archiving [12].As users and enterprises come to rely on diverse sets of data repositories, with variability in service guarantees and underlying hardware integrity, they will require new forms of assurance of the integrity and accessibility of their data. Simple replication offers one avenue to higher-assurance data archiving, but at often unnecessarily and unsustainably high expense.

Modern day cloud storage is based on highly virtualized infrastructure and has the same characteristics as cloud computing in terms of agility, scalability, elasticity and multi-tenancy, and is available both off-premise (Amazon EC2) and on-premise (ViON Capacity Services)[1]. It is believed to have been invented by Joseph Carl Robnett Licklider in the 1960s.[2] However, Kurt Vonnegut refers to a cloud "that does all the heavy thinking for everybody" in his book "Sirens of Titan" published in 1959.[3] Since the sixties, cloud computing has developed along a number of lines, with Web 2.0 being the most recent evolution. However, since the internet only started to offer significant bandwidth in the nineties, cloud computing for the masses has been something of a late developer.

It is difficult to pin down a canonical definition of cloud storage architecture, but object storage is reasonably analogous. Cloud storage services like Open Stack and Sonian Inc., cloud storage products like EMC Atmos and Hitachi Cloud Services, and distributed storage research projects like OceanStore[4] or VISION Cloud are all examples of object storage and infer the following guidelines.

1.3 Cloud Security

Cloud computing security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing.

2. Related Work

D.L.G.Filho and P.S.L.M.Barreto2006 proposed a first display of a secure homomorphic hash function is due to Krohn, Freedman and Mazires[16]. Their function is mostly satisfactory, despite performance issues. The same parameter set can be applied to differently-sized messages. Just as a matter of choice, it is interesting to know that a second construction exists, based on a different hard problem (namely factoring), even if it sports the same characteristics and performance.

The protocol has one main advantage: public keys in their protocol are as large as the data being protected, while the protocol’s public key is just an RSA modulus. Also, the protocol is slightly more flexible, as it does not fix the message size for a given parameter set, and is arguably simpler and more elegant. On the other hand, their proposal may have better performance when elliptic curve groups are employed. In the term of feature set and performance differences, having a second construction with similar properties, but based on a different hard problem, is good for diversity. The main drawback on this method is poor performance.

M.A.Shah et al, (2007) proposed a online service oriented economy (OSOE) in which businesses and end users purchase IT services from a variety of online service providers (OSPs). For this nascent economy to become established, customers will need ways to assess risk and gain trust in OSPs [18, 19].

Third-party auditing is an accepted method for establishing trust between two parties with potentially different incentives [20]. Auditors assess and expose risk, enabling customers to choose rationally between competing services. Over time, a system that includes auditors reduces risk for customers: when combined with a penalty or incentive mechanism, auditing gives incentives to providers to improve their services. Penalty and incentive mechanisms become supportable when risks are well understood. Auditing of OSPs is not feasible yet. First, customers are not yet sophisticated enough to demand risk assessment. Second, OSPs do not yet provide support for third party audits.

A simple method for auditing data integrity is to sample stored data through the public read and write interfaces of a storage service. This approach requires no modification to the service. The auditor simply creates some .fake. User accounts, uploads content, and periodically extracts all of the uploaded content and ensures the result matches the original. *Two approaches need to audit they are external auditing and internal auditing.* Servers need both internal and external audits of OSPs. External audits can only confirm past behavior, so without internal audits, the server could not predict upcoming problems or assess risk exposure. On the other hand, internal audits might not be exhaustive and might be based on incomplete failure models; server can use external audit results to assess whether internal audits are really working.

J.Hendricks et al, (2007) proposed that the server uses erasure coding for secure data retrieval. Unfortunately, erasure coding creates a fundamental challenge: determining if a given fragment indeed corresponds to a specific original block. If this is not ensured for each fragment, then reconstructing from different subsets of fragments may result in different blocks, violating any reasonable definition of data consistency. Systems in which clients cannot be trusted to encode and distribute data correctly use one of two approaches. In the first approach, servers are provided the entire block of data, allowing them to agree on the contents and generate their own fragments.

So cloud develops a new approach, in which each fragment is accompanied by a set of fingerprints that allows each server to independently verify that its fragment was generated from the original value. The key insight is that the coding scheme imposes certain algebraic constraints on the fragments, and that there exist homomorphic fingerprinting functions [21] that preserve these constraints. Servers can verify the integrity of the erasure coding as evidenced by the fingerprints, agreeing upon a particular set of encoded fragments without ever needing to see them. Thus, the two common approaches described above could be used without the bandwidth or computation overheads, respectively.

The fingerprinting functions belong to a family of universal hash functions, chosen to preserve the underlying algebraic constraints of the fragments. A particular fingerprinting function is chosen at random with respect to the fragments being fingerprinted. This "random" selection can be deterministic with the appropriate application of a cryptographic hash function. If data is represented carefully, the remainder from division by a random irreducible polynomial or the evaluation of a polynomial at a random point preserves the needed algebraic structure [22]. The resulting fingerprints are secure, efficient, and compact.

G.Ateniese et al, (2008) proposed the concept of third-party data warehousing and, more generally, data outsourcing has become quite popular. Outsourcing of data essentially means that the data owner (client) moves its data to a third-party provider (server) which is supposed to – presumably for a fee – faithfully store the data and make it available to the owner (and perhaps others) on demand. Appealing features of outsourcing include reduced costs from savings in storage, maintenance and personnel as well as increased availability and transparent up-keep of data.

The problem of Provable Data Possession (PDP) [23, 24] –is also sometimes referred to as Proof of Data Retrievability (POR)– has popped up in the research literature. The central goal in PDP is to allow a client to efficiently, frequently and securely verify that a server – who purportedly stores client's potentially very large amount of data – is not cheating the client. In this context, cheating means that the server might delete some of the data or it might not store all data in fast storage, e.g., place it on CDs or other tertiary off-line media. It is important to note that a storage server might not be malicious; instead, it might be simply unreliable and lose or inadvertently corrupt hosted data. An effective PDP technique must be equally applicable to malicious and unreliable servers. The problem is further complicated by the fact that the client might be a small device (e.g., a PDA or a cell-phone) with limited CPU, battery power and communication facilities.

M.A.shah et al, (2008) describe the study of deployed large-scale storage systems show that no storage service can be completely reliable; all have the potential to lose or corrupt customer data. Today, a customer that wants to rely on these services must make an uneducated choice. He has only negative newsworthy anecdotes on which to base his decision, and service popularity or "brand name" is not a positive indicator of reliability. To know if his data is safe, he must either blindly trust the service or laboriously retrieve the hosted data every time he wants to verify its integrity, neither of which is satisfactory.

Unfortunately, to date, there are no fair and explicit mechanisms for making these services accountable for data loss. The proposed solution to provide storage service accountability is through independent, third party auditing and arbitration. The customer and service enter into an agreement or contract for storing data in which the service provides some type of payment for data loss or failing to return the data intact, e.g. free prints, refunds, or insurance. In such an agreement, the two parties have conflicting incentives.

The service provider, whose goal is to make a profit and maintain a reputation, has an incentive to hide data loss. On the other hand, customers are terribly unreliable, e.g. casual home users. Customers can innocently (but incorrectly) or fraudulently claim loss to get paid. Thus, the proposed scheme involves an independent, third party to arbitrate and confirm whether stored and retrieved data is intact.

A straightforward solution for maintaining privacy during audits is for the customer to encrypt his contents using symmetric-key encryption and keep those keys intact and secret from uninvited parties. Then, the auditor can use existing provably secure, challenge-response schemes on the encrypted contents. This solution is unsatisfactory because an unsophisticated customer is increasingly likely over time either to lose the keys and be unable to recover the contents, or to leak the keys. The solution is to shift the burden of keeping these secret keys to a storage service [25, 26, 27]. Since services are already in the business of maintaining customers' data and privacy, the keys are safer with them. Keeping the data content private from the service is optional. A customer can keep the keys and encrypted data with the same service, thereby revealing the contents to that service and allowing it to provide value-added features beyond storage like search. Otherwise, the customer can separate the keys and encrypted data onto non-colluding services to maintain complete privacy. The auditor is responsible for auditing and extracting both the encrypted data and the secret keys. The protocols, however, never reveal the secret key to the auditor.

K. D. Bowers et al, (2009) proposed the concept of cloud that trend toward loosely coupled networking of computing resources, is unsecure data from local storage platforms. Users today regularly access files without knowing—or needing to know—on what machines or in what geographical locations their files reside. They may even store files on platforms with unknown owners and operators, particularly in peer-to-peer computing environments. While cloud computing encompasses the full spectrum of computing resources, so users focus on *archival* or *backup* data, large files subject to infrequent updates. While users may access such files only sporadically, a demonstrable level of availability may be required contractually or by regulation.

Financial records, for instance, have to be retained for several years to comply with recently enacted regulations. Juels and Kaliski (JK)[30] recently proposed a notion for archived files that they call a *proof of retrievability* (POR). A POR is a protocol in which a server/archive proves to a client that a target file *F* is intact, in the sense that the client can retrieve all of *F* from the server with high probability. In a na¨ıve POR, a client might simply download *F* itself and check an accompanying digital signature. JK and related constructions adopt a challenge-response format that achieves much lower (nearly constant) communication complexity—as little as tens of bytes per round in practice.

JK offer a formal definition of a POR and describe a set of different POR designs in which a client stores just a single symmetric key and a counter. Their most practical constructions, though, support only a limited number of POR challenges. Shacham and Waters (SW) offer an alternative construction based on the idea of storing homomorphic block integrity values that can be aggregated to reduce the communication complexity of a proof. Its main advantage is that, due to the underlying block integrity structure, clients can initiate and verify an unlimited number of challenges.

C. Erway et al, (2009) proposed that the server provides a definitional framework and efficient constructions for *dynamic provable data possession* (DPDP) [32, 33, 34], which extends the PDP model to support provable *updates* on the stored data. Given a file F consisting of n blocks, an update is either insertion of a new block (anywhere in e file, not only append), or modification of an existing block, or deletion of any block. Therefore the update operation describes the most general form of modifications a client may wish to perform on a file. The DPDP solution is based on a new variant of authenticated dictionaries, where we use *rank* information to organize dictionary entries. It results to support efficient authenticated operations on files at the block level, such as authenticated insert and delete.

The operation shows how to extend our construction to support data possession guarantees of a hierarchical file system as well as file data itself. To the best of our knowledge, this is the first construction of a provable storage system that enables efficient proofs of a whole file system, enabling verification at different levels for different users (e.g., every user can verify her own home directory) and at the same time not having to download the whole data use a modified authenticated skip list data structure. This new data structure, called as *rank-based authenticated skip list [35]*, is based on authenticated skip lists but indexes data in a different way. This would perfectly work for the static case. But in the dynamic case, the file system would need an authenticated red-black tree, and unfortunately no algorithms have been previously presented for rebalancing a Merkle tree while efficiently maintaining and updating authentication information (except for the three-party model, e.g.,). Yet, such algorithms have been extensively studied for the case of the authenticated skip list data structure the introduction of authenticated skip lists is presented before the new data structure.

An authenticated skip list to check the integrity of the file blocks. However, this data structure does not support efficient verification of the indices of the blocks, which are used as query and update parameters in the DPDP scenario. The updates in the DPDP scenario are insertions of a new block after the i^{th} block and deletion or modification of the i^{th} block (there is no search key in this case, in contrast to [26], which basically implements an authenticated dictionary).

W. Wang et al, (2009) proposed the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. What is more serious is that for saving money and storage space the service provider might neglect to keep or deliberately delete rarely accessed data files which belong to an ordinary client. Consider the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files.

In order to solve this problem, many schemes are proposed under different systems and security models. In all these works, great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrieve ability of data, etc. The verifier in the model, all the schemes includes two categories: private verifiability and public verifiability. Although schemes with private verifiability can achieve higher scheme efficiency, public verifiability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources [31].

In the cloud, the clients themselves are unreliable or cannot afford the overhead of performing frequent integrity checks. Thus, for practical use, it seems more rational to equip the verification protocol with public verifiability, which is expected to play a more important role in achieving economies of scale for Cloud Computing. Moreover, for efficiency consideration, the outsourced data themselves should not be required by the verifier for the verification purpose.

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., through block modification, deletion and insertion. 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 updates has received limited attention in the data possession applications so far. Moreover, as will be shown later, the direct extension of the current provable data possession (PDP) or proof of retrievability (PoR) 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 verifiability and the supporting of data dynamics for cloud data storage, a framework and an efficient construction for seamless integration of these two components in this protocol design. The contributions are (1) a general formal PoR model with public verifiability for cloud data storage, in which block less verification is achieved; (2) the proposed PoR construction with the function of supporting for fully dynamic data operations, especially to support block insertion, which is missing in most existing schemes; (3) the security of the proposed construction and justify the performance of the scheme through concrete implementation and comparisons with the state-of-the-art.

C.Wang et al, (2012) describes that one fundamental aspect of this paradigm shifting is that data is being centralized or outsourced to the Cloud. From users' perspective, including both individuals and IT enterprises, storing data remotely to the cloud in a flexible on-demand manner brings appealing benefits: relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc

While Cloud Computing makes these advantages more appealing than ever, it also brings new and challenging security threats towards users' outsourced data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing is actually relinquishing user's ultimate control over the fate of their data [36]. As a result, the correctness of the data in the cloud is being put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity. Examples of outages and security breaches of noteworthy cloud services

appear from time to time. Secondly, there do exist various motivations for CSP to behave unfaithfully towards the cloud users regarding the status of their outsourced data. For examples, CSP might reclaim storage for monetary reasons by discarding data that has not been or is rarely accessed, or even hide data loss incidents so as to maintain a reputation.

In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed.

However, the fact that users no longer have physical possession of data in the cloud prohibits the direct adoption of traditional cryptographic primitives for the purpose of data integrity protection. Hence, the verification of cloud storage correctness must be conducted without explicit knowledge of the whole data files, [18], [25]. Meanwhile, cloud storage is not just a third party data warehouse. The data stored in the cloud may not only be accessed but also be frequently updated by the users [8], [16], [17], include insertion, deletion, modification, appending, etc. Thus, it is also imperative to support the integration of this dynamic feature into the cloud storage correctness assurance, which makes the system design even more challenging. Last but not the least, the deployment of cloud computing is powered by data centers running in a simultaneous, cooperated, and distributed manner. It is more advantages for individual users to store their data redundantly across multiple physical servers so as to reduce the data integrity and availability threats. Thus, distributed protocols for storage correctness assurance will be of most importance in achieving robust and secure cloud storage systems. However, such important area remains to be fully explored in the literature. Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works under different system and security models [18], [25], [23], [31], [32], [30], [33], [36].

These techniques, while can be useful to ensure the storage correctness without having users possessing local data, are all focusing on single server scenario.. Although direct applying these techniques to distributed storage (multiple servers) could be straightforward, the resulted storage verification overhead would be linear to the number of servers. As an complementary approach, researchers have also proposed distributed protocols for ensuring storage correctness across multiple servers or peers. However, while providing efficient cross server storage verification and data availability insurance, these schemes are all focusing on static or archival data. As a result, their capabilities of handling dynamic data remains unclear, which inevitably limits their full applicability in cloud storage scenarios.

Quan wang et al, 2012 proposed a erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability against Byzantine servers, where a storage server may fail in arbitrary ways. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization.

The major drawback in this paper is the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, this paper proposes an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. In this proposed scheme rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. There are lot of researchers who have made their contribution towards the evolution of cloud computing and its associated challenges. A consolidated summary of the research of few of them are listed below in the tabular column.

3. Methodology

The proposed system is an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of users' data in the cloud. The proposed scheme concentrates on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphism token with distributed verification of erasure-coded data, this scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, this scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s). This work is among the first few ones in this field to consider distributed data storage in Cloud Computing.

This contribution can be summarized as the following three aspects:

1) Compared to many of its predecessors, which only provide binary results about the storage state across the distributed servers, the challenge-response protocol in this work further provides the localization of data error.
2) Unlike most prior works for ensuring remote data integrity, the new scheme supports secure and efficient dynamic operations on data blocks, including: update, delete and append.
3) Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

3.1 Distributed Cloud Storage

Cloud Storage is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. Hosting companies operate large data centers; and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs. The data center operators, in the background, virtualizes the resources according to the requirements of the customer and expose them as virtual servers, which the customers can themselves manage. Physically, the resource may span across multiple servers. The distributed storage holds good for all types of file format viz. .txt, .pdf, .img, etc. to be stored in a distributed manner in the cloud server. Although the time taken to retrieve would differ for different file formats, the third party auditor is unbiased and works seamlessly in ensuring the data integrity insurance across the cloud server.

Algorithm for Updating and Deleting Data Present in CSS for Multi-Client Environment

- Start
- Cloud Storage Server and Third party auditor Synchronizing Clients and Gets Connected according to their priorities.
- Client generates new Hash tree then sends it to Cloud Storage Server
- Cloud Storage Server updates F and computes new root R'. Runs Exec Update algorithm
- Cloud Storage Server sends old root and new root to client.
- Client first verifies old root value to check whether CSS is updating the same block or not. Runs Verify Update algorithm.
- Client computes new R and verifies the update block. If it fails outputs FALSE.
- Stop

3.2 Digital Signature

A **digital signature** or **digital signature scheme** is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering.

Digital signatures employ a type of asymmetric cryptography. For messages sent through a non secure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital signatures are equivalent to traditional handwritten signatures in many respects; properly implemented digital signatures are more difficult to forge than the handwritten type. Digital signature schemes in the sense used here are cryptographically based, and must be implemented properly to be effective. Digital signatures can also provide non-repudiation, meaning that the signer cannot successfully claim they did not sign a message, while also claiming their private key remains secret; further, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is valid nonetheless.

3.3 Data Security Model

Following mathematical equations presents the data security in cloud computing:

$Df=C(NameNode)$;
$Kf=f * Df$;
C(.): the visit of nodes;
Df: the distributed matrix of file f;
Kf: the stste of data distribution in datanodes;
F: file, file f can be described as;
$F=\{F(1), F(2), ...F(n)\}$, means f is the set of n file blocks.
$F(i) \wedge F(j)=\Phi$, $i!= j$; $j€ 1, 2,3,....n$;
Df is a zero-one matrix, it is L*L, L is the number of datanode.
Thus in order to provide the data security in cloud computing we are presented the below approaches:
$Df' = CA\ (NameNode)$;
$Df = M.Df'$;
$Kf = E(f) * Df$;
CA(.) : authentic visit to namenode;
Df : private protect model of file the distributed matrix;
M : resolve private matrix;
E(f) : encrypted file f block by block, get the encrypted file vector

The above mathematical equation ensures that the user requesting for the data is indeed the owner of the data by stringent verification measures to check for any possible tampering or forging.

3.4 Third Party Auditor

TPA in possession of the public key can act as a verifier, 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.

Table 4.1: Showcases of Single and Multiple audits

Number of attempts	*Single audit*	*Multiple audit*
1	9	2
2	42	1
….	….	….
….	….	….
12	4	8
13	8	2
….	….	….
….	….	….
38	7	1
39	5	0

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.

Data Integrity Verification Algorithm for Multi-Client Environment.

- Start
- Multiple Clients synchronizing connection with Third party auditor as well as Cloud Storage Server.
- Client gets connected with Third party auditor for Request processing.
- Client generates a tag for each file block using signature generation algorithm
- A Merkle Hash Tree is constructed for each file block.
- The root R of the Merkle Hash Tree is signed using the secret key
- Client advertise file, set of signatures and computed root value to the server and deletes it from its local storage
- TPA generates a challenge and sends to the server
- Server generates a proof based on challenge, the proof contains auxiliary authenticate information using Generate Proof algorithm
- The server sends the generated Proof P to the client.
- The Third party auditor validates the proof by generating the root R, using verify Proof algorithm.
- After verification, the Third party auditor can determine whether the integrity is reached.
- Stop

Dynamic data operations support to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The above algorithm involves the third party auditor in synchronizing multiple clients with that of the cloud server. The third party auditor receives the data request from the clients in the form of a challenge by means of a set of signatures, file location and relevant details. The auditor then processes the challenge in the cloud server and transmits the results to the client upon signature and data request authentication. This approach of auditing ensures that the data does not end up to anyone intended other than the client.

The design should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation. The time difference in single and multiple audits to retrieve the file is listed in the above table. The proposed methodology addresses all the critical aspects of data security in cloud computing by incorporating distributed cloud storage, digital signature and usage of third party auditor(s) for single and multiple audits to ensure the correctness of data in the cloud with considerable reduction in time taken to retrieve the data using indigenously built digital signature. The *modus operandi* of this system has three components that are highly inter-connected to ensure the veracity of the data being stored and retrieved by the clients in the cloud server. The three components of this system as described above are as follows:

1. Distributed Cloud Storage
2. Digital Signature
3. Third Party Auditor(s)

These components work in tandem to deliver an efficient and the best possible solution towards data security in the cloud servers. These components are built as per the demands of the industry that is engulfed in complicated issues as more and more clients are raising doubts over storing their data in the cloud due to high levels of insecurity that is prevalent in the remote data centers all across the globe.

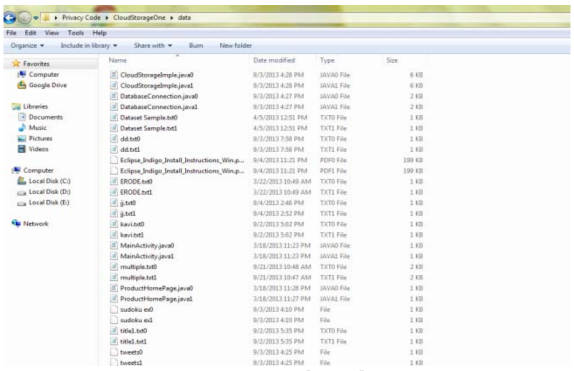
Although bypassing the existing methodology to address the data security challenge would prove to be a quick fix solution, the long term need would remain unfulfilled. Thus, the proposed methodology is not a quick fix solution as it is built on state-of-the-art techniques to face these challenges and will stand testimony in the foreseeable future in cloud computing space. This method of ensuring data integrity correctness assurance has undergone various testing before being proposed as an alternative to the existing and rapidly outgoing system of security in cloud servers. The results of these tests are discussed in the next chapter.

4. Evaluation of Result

The experiment is conducted using Java on a system with an Intel dual core processor running at 2.4 GHz, 1 GB RAM. During the implementation the proposed system made three parts which are major components of proposed approach such as clients, third party auditor and cloud server in order to store client's data. A sample set of file blocks is processed and stored at server. The integrity verification is carried out using different set of challenges and verifies the proof generated by the server. The storage space for key generation, Merkle tree construction and signature generation take place in the constant order. In the integrity verification stage, the proof generation and verification take place in the O (log n) as the file block (mi) is used instead of index. Hence file index recomputation is avoided at each proof generation and verification process.

4.1 Result of Distributed Storage

The underlying principle of distributed storage is to break the data into multiple sequences and store in the server. This distribution can be customized based on the number of partitions that need to be created for storing the data. The user file is distributed to the number of partitioned storage in server with equal memory. Any data request by the user has to pass through the verification procedure for successful data retrieval.



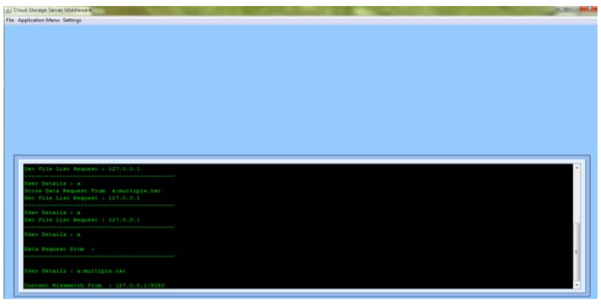
```
Name                                        Date modified         Type         Size
CloudStorageImple.java0                     9/3/2013 4:28 PM      JAVA0 File     6 KB
CloudStorageImple.java1                     9/3/2013 4:28 PM      JAVA1 File     6 KB
DatabaseConnection.java0                    9/3/2013 4:27 PM      JAVA0 File     2 KB
DatabaseConnection.java1                    9/3/2013 4:27 PM      JAVA1 File     2 KB
Dataset Sample.txt0                         4/5/2013 12:51 PM     TXT0 File      1 KB
Dataset Sample.txt1                         4/5/2013 12:51 PM     TXT1 File      1 KB
dd.txt0                                     9/3/2013 7:58 PM      TXT0 File      1 KB
dd.txt1                                     9/3/2013 7:58 PM      TXT1 File      1 KB
Eclipse_Indigo_Install_Instructions_Win.p...  9/4/2013 11:21 PM   PDF0 File    199 KB
Eclipse_Indigo_Install_Instructions_Win.p...  9/4/2013 11:21 PM   PDF1 File    199 KB
ERODE.txt0                                  3/22/2013 10:49 AM    TXT0 File      1 KB
ERODE.txt1                                  3/22/2013 10:49 AM    TXT1 File      1 KB
jj.txt0                                     9/4/2013 2:46 PM      TXT0 File      1 KB
jj.txt1                                     9/4/2013 2:52 PM      TXT1 File      1 KB
kavi.txt0                                   9/2/2013 5:02 PM      TXT0 File      1 KB
kavi.txt1                                   9/2/2013 5:02 PM      TXT1 File      1 KB
MainActivity.java0                          3/18/2013 11:23 PM    JAVA0 File     1 KB
MainActivity.java1                          3/18/2013 11:23 PM    JAVA1 File     1 KB
multiple.txt0                               9/21/2013 10:48 AM    TXT0 File      2 KB
multiple.txt1                               9/21/2013 10:47 AM    TXT1 File      2 KB
ProductHomePage.java0                       3/18/2013 11:28 PM    JAVA0 File     1 KB
ProductHomePage.java1                       3/18/2013 11:27 PM    JAVA1 File     1 KB
sudoku ex0                                  9/3/2013 4:10 PM      File           1 KB
sudoku ex1                                  9/3/2013 4:10 PM      File           1 KB
title1.txt0                                 9/2/2013 5:35 PM      TXT0 File      1 KB
title1.txt1                                 9/2/2013 5:35 PM      TXT1 File      1 KB
tweets0                                     9/3/2013 4:25 PM      File           1 KB
tweets1                                     9/3/2013 4:25 PM      File           1 KB
```

Figure 5.1: Distributed Storage

The above figure represents the outcome of distributed data storage where the data is distributed in multiples of two. In the proposed scheme the distributed storage is partitioned into two. So the size of the user(s) file is equally distributed.

4. 2 Result of Verification

The verification process is to check that the data stored in the server is secure and devoid of any hacking. This is ensured by displaying the result after the signature authentication procedure performed by the proposed system.


Cloud Storage Server Middleware
File Application Menu Settings

```
Get File List Request : 127.0.0.1
------------------------------------------
User Details : a
Store Data Request From  a:multiple.txt
Get File List Request : 127.0.0.1
------------------------------------------
User Details : a
Get File List Request : 127.0.0.1
------------------------------------------
User Details : a

Data Request From  :
------------------------------------------

User Details : a:multiple.txt

Content Mismatch From  : 127.0.0.1:8083
```

Figure 5.2: Verification Result

Figure 5.2 shows the verification taken place in the middleware. If the data is modified in the server by means of hacking, the middleware verifies the signature of the authorized user and check the original. By this verification the middleware displays the result that the content is mismatched or not. Any case of mismatching in the procedure is reported by the verification and thereby leaving no error for hacking or potential data compromise to happen in the server. This verification procedure is conducted in the middleware.

4.3 Result of Performance Analysis

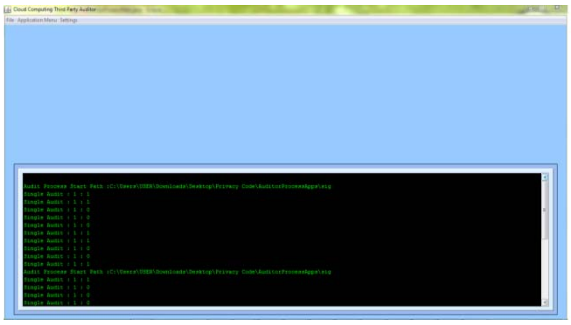
The performance of distributed storage is gauged by the retrievability factor of the server when a data request is put forth by the user(s). There are two types of audits conducted to measure the performance of the server viz. single audit and multiple audits. The primary purpose of auditor(s) is to verify the authenticity of the signature and the correctness of the data requested by the users before the auditor(s) start processing the request.

4.3.1comparison of Single and Multiple Audits

A brief comparison of how single and multiple audits fare in the context of time taken to retrieve the data from the server based on single or multiple requests by the user(s).

4.3.2 Single audit

The single audit process engages only one TPA to verify the authenticity of the signature and the correctness of the data requested by the user before processing the data. The time taken to retrieve the data in case of a single audit process is longer when more than one user requests the data from the server.


Cloud Computing Third Party Auditor
File Application Menu Settings

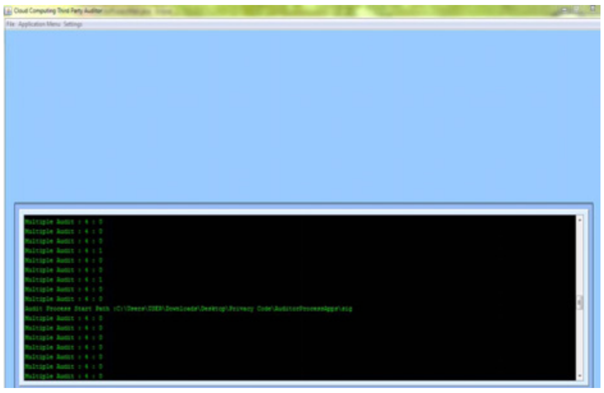
```
Audit Process Start Path :C:\Users\USER\Downloads\Desktop\Privacy Code\AuditorProcessApps\sig
Single Audit : 1 : 1
Single Audit : 1 : 1
Single Audit : 1 : 0
Single Audit : 1 : 0
Single Audit : 1 : 0
Single Audit : 1 : 1
Single Audit : 1 : 1
Single Audit : 1 : 0
Single Audit : 1 : 0
Single Audit : 1 : 1
Audit Process Start Path :C:\Users\USER\Downloads\Desktop\Privacy Code\AuditorProcessApps\sig
Single Audit : 1 : 1
Single Audit : 1 : 0
Single Audit : 1 : 0
Single Audit : 1 : 0
```

Figure 5.3: Single audit

The screen print above indicates the retrieval time for a single audit process. . In the above figure, the second column after single audit indicates the number of auditor(s) particular to this audit, and the sum of all the numeric values in the last column indicates the time taken in seconds to retrieve the data.

4.4 Multiple Audits

The multiple audits process engages more than one TPA to verify the signature and the correctness of the data requested by the user(s) before processing the data. The time taken to retrieve the data in case of a multiple audit process is considerably reduced as more audits are conducted for multiple requests by the users.


Cloud Computing Third Party Auditor
File Application Menu Settings

```
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 1
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 1
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Audit Process Start Path :C:\Users\USER\Downloads\Desktop\Privacy Code\AuditorProcessApps\sig
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
Multiple Audit : 4 : 0
```

Figure 5.4 Multiple audit

The screen print above indicates the retrieval time for a multiple audit process. In the above screenshot, the second column after single audit indicates the number of auditor(s) particular to this audit, and the sum of all the numeric values in the last column indicates the time taken in seconds to retrieve the data. The results discussed above clearly indicate a significant improvement in the way data would be stored in the cloud servers in comparison to the existing storage type. Though the system uses distributed storage technique to store the data, the data security, integrity insurance and correctness assurance is not compromised at any cost owing to the fact that the data verification using the state-of-the-art digital signature is implemented. Also, the time taken to retrieve the data is considerably reduced explains the capability of multiple audits and thereby pushing single audits into oblivion.

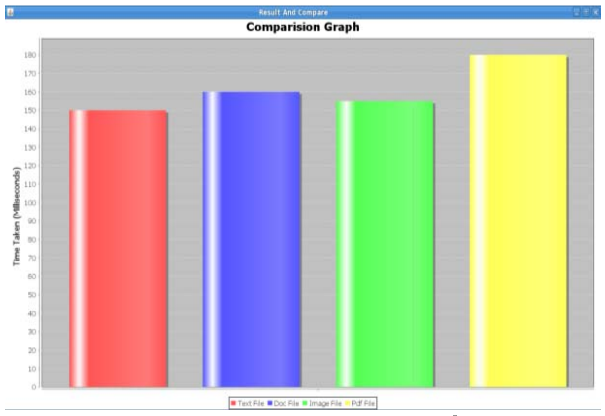


Figure 5.6: Comparison graph

The above graph indicates the time difference between the single and the multiple audition.the performance of multiple audition is far better than the single audition in different type of files such as text ,pdf,etc.,.. This integrated system of distributed cloud storage, digital signature and third party auditor in cloud storage server not only addresses the data storage correctness challenge but also provides a more sophisticated framework that is expected to be accepted by one and all among the cloud computing fraternity.

5. Conclusion and Future Work

To ensure cloud data storage security, it is critical to enable a third party auditor (TPA) to evaluate the service quality from an objective and independent perspective. Public audit ability also allows clients to delegate the integrity verification tasks to TPA while they themselves can be unreliable or not be able to commit necessary computation resources performing continuous verifications.

Another major concern is how to construct verification protocols that can accommodate *dynamic* data files. The public audit ability and data dynamics for remote data integrity check in Cloud Computing. The construction is deliberately designed to meet these two important goals while efficiency being kept closely in mind. To achieve efficient data dynamics, the proposed system improve the existing proof of storage models by manipulating the classic Merkle Hash Tree (MHT) construction for block tag authentication is to support efficient handling of multiple auditing tasks.

In the future work technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

References

[1] Alistair Wyse. PlusNet: Email Update 3rdAugust.http://usertools.plus.net/status/archive/1154603560.htm, August 2006.
[2] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz A. Barroso. Failure trends in a large disk drive population. In USENIX Conference on File and Storage Technologies (FAST 2007), pages 17–29, 2007
[3] David Lazarus. Precious Photos Disappear. San Francisco Chronicle, http://www.sfgate. com/cgi bin/article.cgi?file=/chronicle/archive/2005/02/02/BUG 7QB3U0S1.DTL, February 2005.
[4] JohnKubiatowicz et al. OceanStore: An Architecture for Global-Scale Persistent Storage.ASPLOS, November 2000
[5] Evan Hanson. Hotmail incinerates customer files. CNET News.com, http://news.com.com/Hotmail+incinerates+customer+files/2100-1038_3-5226090.html, June 2004.
[6] Lakshmi N. Bairavasundaram, Garth R. Goodson, Shankar Pasupathy, and Jiri Schindler.An analysis of latent sector errors in disk drives. In SIGMETRICS'07, pages 289–300, 2007.
[7] Lakshmi N. Bairavasundaram, Garth R. Goodson, Bianca Schroeder, Andrea C. Arpaci- Dusseau, and Remzi H. Arpaci-Dusseau.An analysis of data corruption in the storage stack.In USENIX conference on File and Storage Technologies (FAST '08), pages 223–238, 2008.
[8] Muthitacharoen, R. Morris, T. M. Gil, and B. Chen. Ivy:A read/write peer-to-peer file system. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, Dec. 2002.
[9] Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01), pages 188–201,
[10] Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels,R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer,C. Wells, and B. Zhao. OceanStore: An architecture for global-scale persistent storage. In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Cambridge, MA, Nov. 2000. ACM.
[11] Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard.A cooperative Internet backup scheme. In Proceedings of the 2003 USENIX Annual Technical Conference, pages 29–42, San Antonio, TX, 2003
[12] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. In Peer-to-Peer Systems II: Second International Workshop, IPTPS 2003, Springer Lecture Notes in Computer
[13] Science 2753, pages 33–44, Berkeley, CA, USA, Feb. 2003.
[14] E. Gabber, J. Fellin, M. Flaster, F. Gu, B. Hillyer, W. T. Ng B. ¨Ozden, and E. Shriver. StarFish: Highly-available block storage. In Proceedings of the Freenix Track: 2003 USENIX Annual Technical Conference, pages 151–163, San Antonio,
[15] TX, June 2003.
[16] V. Goldberg and P. N. Yianilos. Towards and archival intermemory. In Advances in Digital Libraries ADL'98, pages 1–9, April 1998.
[17] Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. R. Douceur, J. Howell, J. R. Lorch, M.

Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment.In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, Dec. 2002. USENIX.

[18] D.L.G. Filho and P.S.L.M. Barreto, Demonstrating Data Possession and Uncheatable Data Transfer," Cryptology ePrint Archive,Report 2006/150, http://eprint.iacr.org, 2006.

[19] Ian Clarke. Freenet – the free network project. http://freenetproject.org.Allmydata Inc. Allmydata. http://allmydata.com.

[20] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditingto Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.

[21] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure Trends in a Large Disk Drive Population. In Proc. FAST, Feb. 2007.

[22] Schroeder and G. Gibson. Disk Failures in the Real World: What does an MTTF of 1,000,000 hours mean to you? In Proc. FAST, Feb. 2007.

[23] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasure-Coded Data," Proc. 26th ACM Symp. Principles of Distributed Computing, pp. 139-146, 2007.

[24] J. L. Carter and M. N. Wegman. Universal classes of hashfunctions (extended abstract). In Proceedings of the 9th ACM Symposium on Theory of Computing, pages 106–112. ACMPress, 1977.

[25] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf.Security and Privacy in Comm. Netowrks (SecureComm '08), pp. 1-10,

[26] 2008.

[27] Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In ACM CCS'07, Full paper available on e-print (2007/243), 2007.

[28] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, http://eprint.iacr.org, 2008.

[29] Lakshmi N. Bairavasundaram, Garth R. Goodson, Shankar Pasupathy, and Jiri Schindler. An analysis of latent sector errors in disk drives. In GMETRICS'07, pages 289–300, 2007.

[30] Lakshmi N. Bairavasundaram, Garth R. Goodson, Bianca Schroeder, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. An analysis of data corruption in the storage stack.In USENIX conference on File and Storage Technologies (FAST '08), pages 223–238, 2008.

[31] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz A. Barroso. Failure trends in a large disk drive population. In USENIX Conference on File and Storage Technologies (FAST 2007),pages 17–29, 2007.

[32] Bianca Schroeder and Garth A. Gibson. Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you? In USENIX Conference on File and Storage Technologies (FAST 2007), pages 1–16, 2007.

[33] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability:Theory and Implementation," Proc. ACM Workshop Cloud Computing Security (CCSW '09), pp. 43-54, 2009.

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

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

[36] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In CCS, pp. 598–609, 2007.

[37] Y. Dodis, S. Vadhan, and D. Wichs. Proofs of retrievability via hardness amplification. In TCC, pp. 109–127, 2009.

[38] H. Shacham and B. Waters. Compact proofs of retrievability.In ASIACRYPT, pp. 90–107, 2008.

[39] Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Trans.Computers, preprint, 2012, doi:10.1109/TC.2011.245.