ISSN:0975-9646

Secure Multi-Cloud Storage Using Elastic Scattered Storage and Scalable Auditing

Durairaj Moorthy[1], Chandrasekar C[2]

[1] *Department of Computer Science, Karpagam University, Coimbatore, Tamilnadu 642120, India*

[2] *Department of Computer Science,Periyar University, Salam, Tamilnadu 636011, India*

***Abstract*— Cloud storage facilitates users to store data remotely and benefits enterprise network or organization by minimizing the complexity of local hardware and software management. Despite the understandable on-demands, an infrastructure based services exposes user physical possession of outsourced data which certainly creates new security threats affecting data accuracy in cloud. In order to address the emerging problem against secure and reliable cloud storage, this paper proposes an elastic scattered storage scalability auditing (ESSSA) method, comprising homomorphic token and distributed scoring coded data. ESSSA method supports reliable user access to cloud storage with better quality communication and reduced computation cost. The auditing outcome not only guarantees robust cloud storage accuracy assurance, but at the same time achieves quick data fault localization through detection of malfunctioning server. The progressive or dynamic feature of cloud data is also manageable by proposed ESSSA method with secure and potential dynamic functioning on outsourced data including block modification, deletion, and addition. Analysis shows the proposed ESSSA method is highly potential and efficient against complex failure and malicious data modification attack. In addition, the better performance of the proposed ESSSA method is evaluated to justify the better quality communication and reduced computation cost. The proposed secure and reliable storage services in cloud based on elastic scattered storage scalability auditing security is implemented in JAVA. Experimental results reveals secure ESSSA mechanism with reliable storage services in cloud and better quality compared to existing methods. Security analysis and performance evaluation in terms of data integrity, communication cost and fault localization prove the better performance of ESSSA mechanism.**

***Keywords* – secure cloud computing, reliable scattered storage, data integrity, fault localization, quality communication, data dynamics**

I. INTRODUCTION

Numerous methods and techniques are rising up the valuable of cloud computing. Cloud computing is growing due to the wide practice of computer technology and frequent use of internet based development. Low-cost but efficient processors involving software as a service (SaaS) computing frameworks are broadcasting data servers into group of computing service on a large scale. The rising network bandwidth and secure network communication in cloud increase the degree of user access with high quality data or software on a remote server.

Broadcasting data into the cloud provides more ease to users as no further care is in need about the burden of direct hardware management. Most of the internet based online applications fails in supporting large quantity of storage space and managing dynamic computing resources. As a result, users are not completely satisfied with the integrity and availability of data. Even though the cloud infrastructures are much more dominant and dependable than personal computing resources, a wide range of both interior and exterior attacks for data integrity still exist. For Instance, cloud security defense to protect cloud computing against http-DoS and XML-DoS attacks [7] still faces exterior attacks for data integrity.

On the other hand, while users may not preserve a local copy of outsourced data, the chance of malfunctions by cloud service providers against cloud users is high extracting the type of outsourced data. For example, to gain the profit scope by reducing cost, service providers omits infrequently accessed data [5]. Correspondingly, service providers try to cover data loss occurrence in order to preserve a reputation [8]. As a result, outsourcing data into the cloud is cost-effectively interesting for large scale data storage, but with a deficient of providing robust guarantee of data integrity and availability.

In order to ensure the cloud data integrity and availability with quality implementation of cloud storage service an efficient method for cloud users is necessary to fulfill organizations data accuracy verification. But, the problems arise in the physical control of data restricting the direct execution of standard cryptographic concepts to achieve data integrity protection. Therefore, the confirmation of cloud storage accuracy is done with implicit knowledge of the whole data files [9]. At the same time, cloud storage is not just a third party data warehouse. The data loaded in the cloud is not only often accessed but also repeatedly modified, with addition or deletion of data by the users [10]. Therefore the challenging relies on the cloud storage correctness assurance. More significant concern is to store data iteratively within different physical servers by individual users in order to reduce security threats like data integrity and availability, still increasing the challenge of cloud storage.

In cloud data storage, a user loads data into a group of cloud servers operating in a shared way with the help of cloud service provider. The basic cloud storage service architecture is depicted below

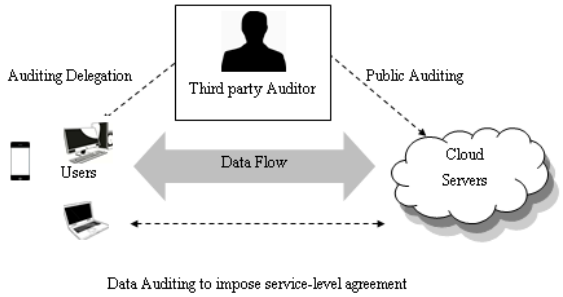

Fig. 1: Cloud Storage Service Architecture

Above figure 1 represents network architecture for cloud storage service. Three different network entities are recognized namely user, cloud server and third party auditor. User depends on the cloud for data storage and computation. The users are probably the enterprise, organization or an individual customer. Cloud server controls the cloud service provider regarding data storage service management providing significant storage space and resources. Third party auditor holds a better ability than user responsible for trusted access and reports the vulnerabilities in storage service.

The point to point communication between each cloud server and user are unauthenticated and unreliable causing threats from external attacks affecting data integrity. A detailed analysis of the security challenges in cloud computing [13] elaborates the existing issues and demand of novel mechanism. For internal attacks, a cloud service provider is selfish, unprotected and probably malicious. For external attacks, the attackers grows outside the control of cloud service provider causing data integrity related threats, data loss, faults, and complex failures. The attackers attempt to learn the data in cloud storage. Proposed ESSSA mechanism is developed in a multi-cloud approach that utilizes two or more cloud services to reduce the difficulty of common data loss. In addition, ESSSA mechanism in multi cloud approach optimizes fault tolerance by adopting homomorphic tokens.

The capability of adversary increases security bugs like external and internal threats against the cloud data integrity. More particularly, the attacker focus in repeatedly corrupting the user's data files stored on cloud servers. Once a server is included in cloud, an adversary corrupts the original data files through modifications. The attempt of data corruption in cloud storage is reduced in ESSSA mechanism through the development of fault localization with the help of pre-compute tokens. Fault localization identifies the misbehaving servers in the multi-cloud, thus reducing the data corruption.

Most of the current secure cloud computing like [2], [4], and [6] for ensuring security related remote data integrity is insignificant due to changeability criteria. The changeability criteria denote modifications of the data files in the cloud storage. On updating data files, the existing security schemes fail in noticing. In contrast to most early works for guaranteeing remote data integrity, the new ESSSA mechanism further offers secure and efficient dynamic operations on data blocks, comprising add, delete and update of data files. Due to the widespread distribution in the accessible cloud services, from the user's perspective become complex to choose the usability of services and on which criteria the selection is to be made. Therefore, user's Quality of Service (QoS) requirements based on ranking [14] supports optimal selection of service. But, still the quality provided to the data is affected in the presence of malicious data modification attack and server colluding attacks. The ESSSA mechanism develops error free server through fault localization with better file recovery. Thus improves data availability against complicated failures, malicious data modification and server colluding attacks increasing the quality.

In order to promise the security and reliable cloud storage, this paper intends in designing an efficient mechanism for dynamic data verification and operation. The proposed mechanism achieves the following objectives:

i. Develop Storage correctness to guarantee users regarding the proper data storage and also the protection against damage throughout the lifetime of data in cloud.
ii. Establish Fault localization to successfully spot the faulty server when data corruption or misbehaving is detected.
iii. Perform Dynamic data operation to preserve the equivalent state of storage accuracy guarantee even if enterprise attempt to add, delete or modify data block files in the cloud.
iv. Better quality communication to facilitate individual customer in achieving storage accuracy guarantee checks with least overhead.

The proposed methodology with the above contribution intends in supporting secure and reliable data storage in multi-cloud. The rest of this paper is organized as follows. Section 2 discusses proposed ESSSA method. Section 3 gives the security analysis and performance evaluations, followed by Section 4 which overviews the related work. Finally, section 5 concludes the proposed work.

II. PROPOSED METHODOLOGY

In general cloud data storage system, the data stored in the cloud by users is not own locally. Hence, ensure the data files correctness and availability being loaded into the multi-cloud server. The major challenge is to potentially identify the unauthorized access to the data because of server malfunctions. Therefore, the primary objective is to identify the inconsistencies caused by the server with data error to minimize storage faults and to facilitate the detection of strong threats. In order to address these problems, Elastic Scattered Storage Scalability Auditing (ESSSA) mechanism is presented to ensure robust multi-cloud data storage.

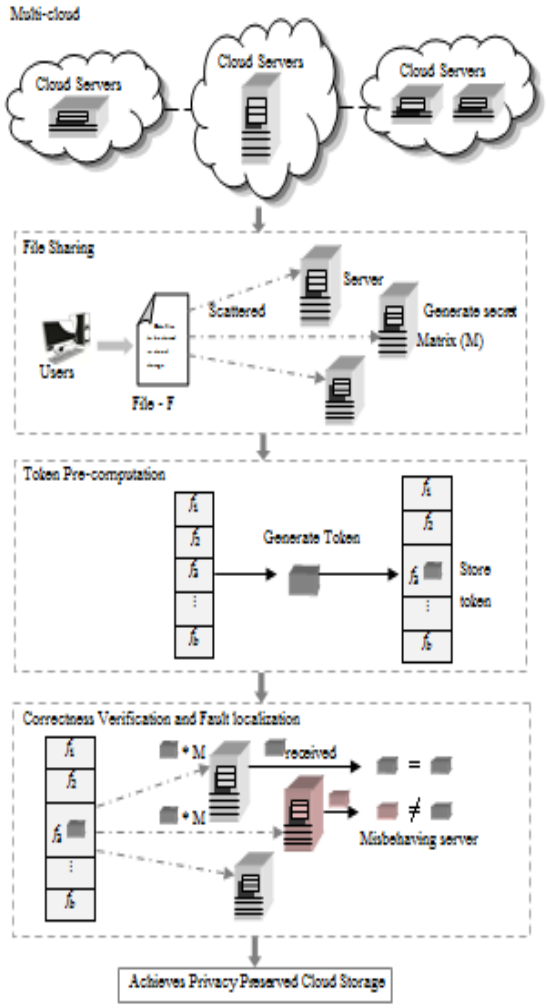

Fig. 2: Overall Architecture of Secure Cloud Storage Based on ESSSA Mechanism

The below figure 2 outlines the overall architecture of robust cloud storage based on ESSSA mechanism. Initially, the ESSSA mechanism elaborates the file sharing across cloud server. Secondarily, homomorphic token is developed. The task of token computation is based on hash functions in order to maintain the homomorphic features which are combined with the verification of scoring coded data in terms of file retrieval and fault recovery. In addition, the mechanism develops a test-response protocol for verifying the storage correctness along with the detection of misbehaving servers. Finally, enhance the ESSSA mechanism to third party auditing.

A. *File Sharing Preparation*

In cloud data storage, ESSSA mechanism scatters the data file F iteratively across a set of b= a + k shared servers. An (b, k) erasure correcting LDPC IRA codes [11] is adopted to generate k iterative parity vectors from b data vectors. The source b data vector are rebuilt from any b out if the b+ k data and parity vectors. On loading each of the b+ k vectors on different servers, the source data file is able to tolerate the failure of any k of the b+ k servers without any data loss, with a space in the clouds of k/b. The unaltered b data file vector along with k parity vector is shared across b+ k different servers.

Let the number of files F collections are $F = (F_1, F_2, \dots, F_b)$ and each file with column vector F_i is denoted as $F_i = (f_{1i}, f_{2i}, \dots, f_{li})^T (i \in \{1, \dots, b\})$with 1 as data vector size in blocks. All these blocks are components of GF (q) as in erasure correcting LDPC IRA codes [11]. The efficient design with parity vectors is obtained using scoring coded with the data scattering matrix M, derived from a $b \times (b + k)$ matrix.

$$\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_b & \beta_{b+1} & \cdots & \beta_n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \beta_1^{b-1} & \beta_2^{b-2} & \cdots & \beta_b^{b-1} & \beta_{b+1}^{b-1} & \cdots & \beta_a^{b-1} \end{pmatrix} \quad (1)$$

Here $\beta_j (j \in \{1, \dots, a\})$ are discrete components arbitrarily chosen from GF(q). After an array of basic row transformations, the desired matrix M is written as

$$M = (I|S) = \begin{pmatrix} 1 & 0 & \cdots & 0 & s_{11} & s_{21} & \cdots & s_{1k} \\ 0 & 1 & \cdots & 0 & s_{21} & s_{22} & \cdots & s_{2k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & s_{b1} & s_{b2} & \cdots & s_{bk} \end{pmatrix} \quad (2)$$

Where I represents the $b \times b$ identity matrix and S denote the secret parity generation matrix with size$b \times k$. By multiplying F by M, the user receives the encoded file G.

$$G = F \cdot M = \left(G^{(1)}, G^{(2)}, \dots, G^{(b)}, G^{(b+1)}, \dots, G^{(a)}\right) = \left(F_1, F_2, \dots, F_b, G^{(b+1)}, \dots, G^{(a)}\right) \quad (3)$$

Here $G^{(j)} = \left(g_1^{(j)}, g_2^{(j)}, \dots, g_l^{(j)}\right)^T (j \in \{1, \dots, a\})$. As equation (3) clearly reproduces the source data file vectors of F through multiplication and the rest part $\left(G^{(b+1)}, \dots, G^{(a)}\right)$ are k parity vectors generated based on F. Depending on distributed scoring coded data in the file sharing preparation provide redundancy parity vectors and guarantee the data reliability.

B. *Token Pre-computation Test*

The ESSSA mechanism completely depends on the pre-computed verification token in order to obtain guarantees of data storage correctness and data fault localization. The concept behind token test is to pre-compute a definite number of verification tokens on distinct vector $G^{(j)} (j \in \{1, \dots, a\})$ before file sharing. Additionally, hide arbitrary subset of data blocks for each token. Afterward, test the cloud servers with a group of arbitrary generated block indices to ensure the users regarding the storage correctness for the data in the cloud. If the users send a test request, each cloud server computes an indication over the distinct blocks and returns them to the user. The values of the indications should match the equivalent tokens pre-computed by the user. At the same time, as all servers functions over the similar subset of the indices, the demanded test-response values for correctness

check must also be a valid key determined by secret matrix S.

The user stores the pre-computed tokens locally to prevent the need for encryption and lower the bandwidth overhead during dynamic data operation like addition, deletion and modifications. The token generation is as follows.

i. Derive an arbitrary test value α_i of GF (q) based on a test key k_t by $\alpha_i = f_{k_t}(i)$ and a permutation key k_p^i based on k_p.
ii. Forecast the set of arbitrary-chosen indices.
iii. Calculate the token as

$$T_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[I_q] \quad (4)$$

Where $G^{(j)}[I_q] = g_{I_q}^{(j)}$ and r denotes random value, j represents the selection server.
Formerly if all tokens are computed, then strengthen each parity blocks $g_i^{(j)}$ before file sharing as below

$$g_i^{(j)} \leftarrow g_i^{(j)} + f_{k_j}(cs_{ij}), i \in \{1, \dots, l\} \quad (5)$$

Where k_j is the secret key for parity vector $G^{(j)}(j \in \{b+1, \dots, a\})$. This is for security of the secret matrix M. At last, users scatter all the encoded vectors $G^{(j)}(j \in \{1, \dots, a\})$ across the cloud servers $CS_1, CS_2, \dots, CS_a$.

C. Correctness Verification and Fault Localization

Fault localization is a necessary condition for eliminating faults in storage systems. In addition, fault identification is also crucial to detect the security bugs from external attacks. But most of the previous mechanism [1], [2], and [3] do not clearly concentrate the problem of data fault localization, just offering outcome for the storage verification. While ESSSA mechanism address the problems by adopting correctness verification and fault localization in means of identifying the misbehaving server into test-response protocol. The response values from servers for each test comprise information to spot robust data faults in addition to the determination of correctness with respect to storage.

More particularly, the process of i-th test-response for a quality assurance over the j servers is describes below

i. The user exposes the α_i as well as the i[th] permutation key $k_p^{(i)}$ to each server.
ii. The server loading vector G(j) (j ∈ {1, . . . , a}) combines those r rows precise by index $k_p^{(i)}$ into a linear combination response $R_i^{(j)}$

$$R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(i)} * k_p^{(i)} \quad (6)$$

iii. Thus, send back $R_i^{(j)}$ as in above equation (5).
iv. With the acceptance of $R_i^{(j)}$ from all the servers, the user takes away strengthen values in $R^{(j)}$ (j ∈ {b+1, . . . , a}).
v. Finally the user verifies whether the received values sustain a valid key determined by secret matrix M.

$$\left(R_i^{(1)}, \dots, R_i^{(b)}\right) \cdot M = \left(R_i^{(m+1)}, \dots, R_i^{(n)}\right) \quad (7)$$

If the above equation (6) holds, the test is passed. Otherwise, test value denotes the existing file block corruptions. With this detection of inconsistencies among the storage, additionally depend on pre-computed verification token to spot the robust data errors. Each response $R_i^{(j)}$ is computed properly similar to token $T_i^{(j)}$. Thus, the user is able to find which server is misbehaving by verifying $R_i^{(j)} = T_i^{(j)}$.

D. File Retrieval and Fault Recovery

By choosing system parameters like random values r, test values t appropriately and performing adequate times of verification, the potential file is retrieved with high probability. Similarly, the difference in pre-computed token and received test-response values ensures the detection of misbehaving servers. The lately recovered blocks are then redistributed to the misbehaving servers to sustain the correctness of storage.

E. Elastic Scattered Storage Scalability Auditing

The principle objective of elastic scattered storage scalability auditing (ESSSA) process should depress the establishing new vulnerabilities towards user data privacy. More specifically, ESSSA should not hear user's data content through the entrusted data auditing. ESSSA design is based on linear property of the parity vector strengthening process. As mentioned before, strengthening process is for protection of the secret matrix M against cloud servers. As a result, if a data vector is strengthening before file sharing then storage verification process is successful providing a privacy-preservation. Finally, the ESSSA design is described as follows

i. Before file sharing, the users strengthen each file block data $g_i^{(j)}$ in $\left(G^{(1)}, \dots, G^{(b)}\right)$ by $g_i^{(j)} \leftarrow g_i^{(j)} + f_{k_j}(cs_{ij})$ as in equation (5).
ii. Based on the strengthened data vector $\left(G^{(1)}, \dots, G^{(b)}\right)$, the users generate k parity vectors $\left(G^{(b+1)}, \dots, G^{(a)}\right)$ through the secret matrix M as generated in equation (3).
iii. The user calculates the i[th] token for server j as in equation (4).
iv. The user sends the token set $T_i^{(j)}$ secret matrix M, permutation and test key $k_p^{(i)}$ and $k_k^{(i)}$ respectively to ESSSA for auditing entrustment.

The correctness validation and misbehaving server identification for ESSSA covers secret strengthening key k_j. As the secret secrete strengthening key k_j (j ∈ {b+1, . . . , a}), the attacker is not able to learn the data content information during auditing process. Hence, the privacy preserving third party auditing is achieved. ESSSA mechanism comprising of file encoding, token pre-computation, and strengthening provides reliable storage

services in cloud. The algorithm for ESSSA mechanism is illustrated below

Algorithm for ESSSA Mechanism

Input: Data File F in cloud server CS

Step-1: //File sharing preparation
- Multiply File F by secret parity matrix M
- Obtain original File

Step-2: //Pre-compute Token
- Choose total block size l, number size n, tokens t, and indices r per verification
- Generate permutation key k_p and test key k_t
  - **For** vector $G^{(j)}$, j =1, n **do**
    - **For** round i =1, t **do**
      - Derive an arbitrary test value $\alpha_i$of GF (q) based on k_t and k_p.
      - Compute token $T_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[I_q]$ as in equation (4).
    - **End For**
  - **End for**
  - Store all the tokens T_i locally
- **End** token Computation

Step-3: //Correctness Verification and Fault localization
- Re-compute arbitrary test value α_i
- Send to all the cloud servers
- Receive from servers $R_i^{(j)}$ as in equation (6)
- **If** the received values sustain a valid key with secret matrix M as in equation (7) **then**
  - Accept and ready for the next test
- **Else**
  - **If** ($R_i^{(j)} != T_i^{(j)}$) **then**
    - **Return** server j is misbehaving
  - **End if**
- **End if**

Step-4: // File Recovery
- Assume block corruption is detected and serve misbehaving is identified
- Load r rows of file blocks from servers and recover the file using scored code
- Resend the recovered files to respective servers.
- **End**

Above ESSSA algorithm illustrate the complete process involved such as file sharing, pre-computation of tokens, storage correctness verification, fault localization in identifying misbehaving servers and file recovery. The level of cloud storage is maintained in ESSSA mechanism even after the user modifies, delete or add additional data files in the cloud.

F. Dynamic Data Operation like Addition, Deletion and Modification

The data stored in cloud are dynamic like electronic documents, images, or log files etc. Hence, the probability of block-level operations like addition, deletion and modification of data file is high with better ensuring of storage accuracy. For performing data dynamic operation, the user produces the equivalent resulted file blocks and parities. Below Figure 3 gives the high level logical representation of data block update or modification, addition and deletion operation.

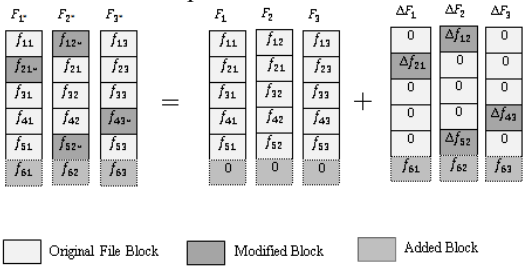

Fig. 3: Logical Operation of Data Dynamic comprising block Update, Add and Delete

Above figure 3 illustrate the logical operation of data dynamics. The Zero components are utilized in the ΔF place to denote the unmodified blocks. The change in data is related to the corresponding modifications in storage verification tokens to hold the alterations on data blocks. The test-response successfully performs the dynamic data operations. For data changes in cloud server, re-compute the whole parity blocks and verification tokens. The data dynamic operations like update, delete, and add are elaborated below.

i) Update or Modification Operation

The process of data update in cloud storage is defined as the modification or update operation from the current value f_{ij} to $f_{ij} + \Delta f_{ij}$. Construct a general update matrix ΔF as

$$\Delta F = (\Delta F_1, \Delta F_2, \dots, \Delta F_m) \quad (8)$$

In order to preserve the equivalent parity vectors and maintain reliable original file, multiply ΔF by M as derived in equation (2). Hence, generate the update information for both the data vectors and parity vectors.

ii) Delete Operation

Certain data needs to be deleted after stored in the cloud. The user replaces the current available data block to zero or pre-defined symbol blocks during delete operation. In order to support delete operation replace the Δf_{ij} in ΔF to be $-\Delta f_{ij}$. Additionally, the entire affected tokens are modified and parity information is updated similar to modification operation.

iii) Add Operation

Sometime users desire to add or extend the size of stored data. The data file is added at the end of the block. Consider the file matrix F as in equation (1) and perform the addition operation. The addition of block needs a revision of the token pre-computation equation (4) for each test-response token by adding i[th] token on server j is modified as below.

$$T_i^{(j)} = \sum_{q=1}^{r_{\max(i)}} \alpha_i^q * G^{(j)}[I_q] \quad (9)$$

Above derived formula ensures that r indices on regular are added into the range of original blocks. When the user is prepared to add new blocks, then corresponding file blocks and the parity blocks are produced. Similarly, the total length of each vector G (j) is also increased. Add the newly generated $T_i^{(j)}$ into old token for addition operation.

III. SECURITY ANALYSIS AND PERFORMANCE EVALUATION

The security analysis focuses on the adversary attacks. ESSSA mechanism is implemented in JAVA. Moreover, evaluate the efficiency of ESSSA mechanism through adoption of both scoring coded data in file sharing and verification token pre-computation. Analyze ESSSA mechanism in terms of data integrity, communication cost and fault localization in comparison to BLS and RSA based instantiations [1], attribute based encryption (ABE) techniques [2] and Nefeli, [3] a hint-based VM scheduler that serves as a gateway to IaaS-clouds.

A. *Data Integrity*

Data integrity is to maintain the cloud storage data in terms of accuracy or correctness guarantees from being altered intentionally, or unintentionally. In ESSSA mechanism, the analysis on correctness verification is decided based on obtained all the response $R_i^{(j)}$ from servers is proper.

TABLE 1: NUMBER OF FILE VS. DATA INTEGRITY

Number of File (MB)	Data Integrity (%)			
	BLS and RSA based instantiations	ABE Technique	Nefeli	ESSSA Mechanism
2	37	41	46	51
4	43	47	51	57
6	56	58	61	67
8	50	54	59	65
10	63	66	70	75
12	77	79	83	88
14	71	73	78	84

Based on the above tabulation (table 1) regarding data integrity, a graph is depicted below

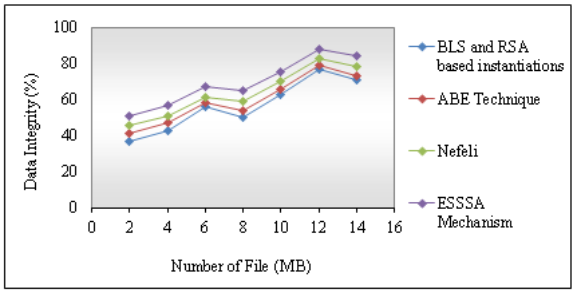

Fig. 4: Number of File vs. Data Integrity

Above figure 4 describes the level of security in multi-cloud storage in terms of data integrity of files. Proposed ESSSA mechanism provides better data integrity of about 17-27 % compared to BLS and RSA based instantiations [1], around 19-13% to ABE techniques [2] and about 9-6% to Nefeli [3]. As users stores data redundantly across multiple physical servers or multi clouds, the data integrity threats is reduced. Moreover, elastic scattered mechanism with explicit dynamic data supports like block update, delete, and add achieves storage assurance cloud data integrity and better quality communication. However, the other existing techniques [1], [2] and [3] falls into the traditional data integrity protection mechanism, where local copy of data are to be stacked for the verification.

B. *Communication Cost*

Communication cost is defined as the reduction of computational overhead on the multi-cloud server, through great probability identification of data corruption. Depending on third party providers of data processing and data transmission services, the communication cost is decided.

TABLE 2: BLOCK SIZE (KB) VS. COMMUNICATION COST IN TIME (SECOND)

Block Size (KB)	Communication Cost in time (second)			
	BLS and RSA based instantiations	ABE Technique	Nefeli	ESSSA Mechanism
50	530	495	481	470
100	393	380	350	322
150	377	362	335	286
200	517	493	447	415
250	449	430	409	359
300	312	298	276	258
350	270	265	241	212

Above tabulation for communication cost depicts the better performance of proposed ESSSA mechanism consuming less time compared to other existing scheme like BLS and RSA based instantiations [1], attribute based encryption (ABE) techniques [2] and Nefeli, [3] a hint-based VM scheduler that serves as a gateway to IaaS-clouds. Based on the above table 2, a graph is depicted below

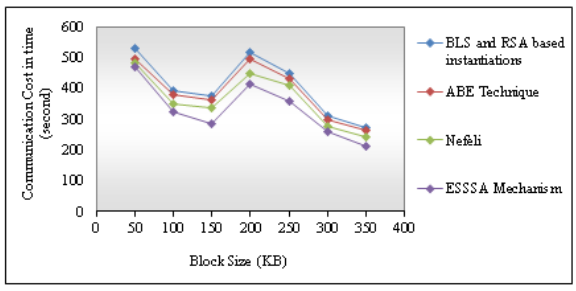

Fig. 5 Block Size (KB) vs. Communication Cost in time (second)

Above figure 5 shows the communication cost taken in storing block files. Proposed ESSSA mechanism offers lower communication cost of about 15-20% compared to existing BLS and RSA based instantiations [1], nearly 5-15% than ABE techniques [2] and about 3-13% than Nefeli [3], the. Because, ESSSA mechanism adopts the homomorphic token with shared verification of scoring-coded data to check the storage overhead and computation cost ensuring better storage accuracy. If the number of detected misbehaving servers is less than the parity vectors, use scoring-coded to recover the corrupted data instantly reducing communication cost. In [1], TPA performs

multiple auditing tasks simultaneously and ABE [2] faces scalability issues causing high storage overhead and communication cost. Even though the time costs of key generation, encryption and decryption processes in ABE [2] are all linear with the number of attributes, storing all those data leads to overhead.

C. *Fault Localization*

Fault localization represents the detection of faults or error found in the data or identification of misbehaving servers for efficient cloud storage.

TABLE 3: NUMBER OF DATA FILE VS. FAULT LOCALIZATION

Number of Data File (MB)	Fault Localization (%)			
	BLS and RSA based instantiations	ABE Technique	Nefeli	ESSSA Mechanism
5	53	49	57	62
10	36	31	39	46
15	31	28	34	41
20	27	25	30	36
25	38	35	44	51
30	43	40	49	54
35	49	44	53	59

Based on the above tabulation (table 3) for fault localization, a graph is depicted below

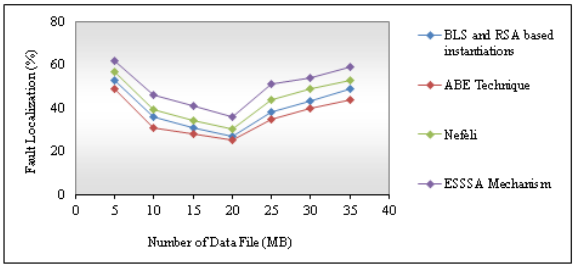

Fig. 5 Number of Data File vs. Fault Localization

Above figure 5 describes the fault localization performance in the presence of data faults or errors in block file. Proposed ESSSA mechanism address the faults more potentially than existing schemes about 15-20% BLS and RSA based instantiations [1], about 20-25% than ABE techniques [2] and approximately 8-15% than Nefeli [3]. Due to inability of deciding the misbehaving servers, the existing schemes struggle in minimizing fault. Whereas proposed ESSSA mechanism provides better fault localization. On comparing received response $R_i^{(j)}$ to secrete matrix M multiplied with $T_i^{(j)}$, the misbehaving server is identified in ESSSA mechanism facilitating better fault localization.

Finally, the ESSSA mechanism proves their better efficiency in terms of high data integrity, lower computational cost and better fault localization ability compared to other existing schemes like BLS and RSA based instantiations [1], attribute based encryption (ABE) techniques [2] and Nefeli, [3] a hint-based VM scheduler that serves as a gateway to IaaS-clouds.

IV. RELATED WORKS

Recent research in data integrity of cloud data storage (CDS) in [15] model ensures the remote data integrity. The CDS model comprise of Multi Agent Systems (MAS) and data encoding to provide security. On one hand the CDS model concentrates on the data integrity, while the security related issues are unsolved. Therefore a secure virtualization in [4] enhances the security of cloud computing by protecting both the integrity of guest virtual machines and the cloud infrastructure components. And this model is called Advanced Cloud Protection System (ACPS), intends at ensuring increased security to cloud resources. More specifically, one of the cloud computing security threats is HTTP-DoS and XML-DoS attacks. In [6], Cloud TraceBack (CTB) is developed to detect the source of these attacks. In addition the use of a back propagation neutral network called Cloud Protector is also prepared to detect and filter such attack traffic.

But all the above [15], [4], and [6] schemes focused on static data. The problem of these schemes lies in preprocessing stages performed by the users at the time of outsourcing the data files. The changeability of files even with few bits results in errors and high computation complexity. The proposed ESSSA mechanism addressed the errors and computation complexity through fault localization and file recovery that verifies token based on test key k_t and permutation key k_p. Additionally, the proposed ESSSA mechanism is also able to perform dynamic operation like add, delete and update with the outsourced data file.

Even the dynamic operations are performed in [1] based on Third party auditing that reduces the participation of the user through the auditing of unbroken data stored in the cloud. In addition [8] also introduces Third party auditing. The establishment of third party auditing [1] detects the potential security threats with complete dynamic data updates. An attribute based encryption (ABE) techniques [2] encrypted each data file to secure outsourced data. Furthermore, Nefeli [3] a virtual infrastructure gateway also improves quality of the rendered services and data availability of cloud storage.

But these mechanism [1], [8], [2], and [3] do not clearly concentrate the problem of data fault localization, just offering outcome for the storage verification. While ESSSA mechanism address the problems by adopting correctness verification and fault localization in means of identifying the misbehaving server through test-response protocol. Furthermore, ESSA mechanism strengthen or append parity block with arbitrary noise, so the malicious servers no longer able to derive the correct secret matrix M increasing the level of security.

Cooperative Provable Data Possession (CPDP) Scheme [12] based on homomorphic verifiable response and online data storage using implicit security [9] also faces high complexity. More specifically, CPDP scheme for large files is affected by the bilinear mapping operations due to its high complexity. By detecting the misbehaving servers in cloud, the ESSSA mechanism reduces the complexity.

Some research studies like Iris [5] concentrate in authenticated file system to reduce workloads for large

organization by storing data in the cloud and allow flexible design against untrustworthy service providers. But, their scheme concentrates on single server scenario and affects data availability scopes against server failures, leading to issues like data file recovery. In contrast, by choosing system parameters like random values r, test values t appropriately and performing adequate times of verification, potential file is retrieved with high probability in proposed ESSSA mechanism. Moreover, the ESSSA mechanism loads the data file in multi-cloud system providing a better data availability.

V. CONCLUSION

The proposed ESSSA mechanism investigates the difficulties faced against data security in multi-cloud data storage. The proposed successful and elastic shared scattered mechanism with explicit dynamic data support comprising block update, add and delete achieves the guarantee of data integrity and availability with reliable multi-cloud storage service for users. Depending on distributed scoring coded data in the file sharing preparation provide redundancy parity vectors and guarantee the data reliability. With the usability of homomorphic token with shared verification of scoring - coded data, ESSSA mechanism ensures high storage accuracy and data fault localization. During the detection of data corruption in storage correctness verification across shared servers, the ESSSA mechanism immediately detects the misbehaving servers. Furthermore, the ESSSA mechanism allow third party auditing in order to reduce time, computation resources utilization and even the load of users. The third party auditing securely entrust the integrity checks regarding cloud storage services. In addition, dynamic operation support like update, add and delete enhance the reliable data storage. Through detailed security analysis and wide experiment results justifies ESSSA mechanism is highly efficient and elastic to complex failure, malicious data modification attack, and even server colluding attacks. The performance parameters such as data integrity, communication cost, and fault localization proves the better performance of ESSSA mechanism compared to other existing schemes.

REFERENCES

[1] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", IEEE Transactions on Parallel and Distributed Systems, Volume:22 , Issue: 5, May 2011.

[2] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou, "Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption", IEEE Transactions On Parallel And Distributed Systems Volume:24 , Issue: 1, Jan. 2013.

[3] Konstantinos Tsakalozos, Mema Roussopoulos, and Alex Delis, "Hint-Based Execution of Workloads in Clouds with Nefeli", IEEE Transactions on Parallel And Distributed Systems, VOL. 24, NO. 7, July 2013.

[4] Flavio Lombardi, Roberto Di Pietro, "Secure virtualization for cloud computing", Elsevier science Direct on Network and Computer Applications, 2011.

[5] Emil Stefanov, Marten van Dijk, Ari Juels and Alina Oprea, "Iris: A Scalable Cloud File System with Efficient Integrity Checks", Annual Computer Security Applications Conference, 2012.

[6] Ashley Chonka, Yang Xiang, Wanlei Zhou, Alessio Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks", Elsevier science Direct on Network and Computer Applications, 2011.

[7] S. Subashini, V.Kavitha, "A survey on security issues in service delivery models of cloud computing", Elsevier science Direct on Network and Computer Applications, 2011.

[8] Dalia Attas and Omar Batrafi, "Efficient integrity checking technique for securing client data in cloud computing", International Journal of Electrical & Computer Sciences IJENS Vol: 11 No: 05, 2011.

[9] Abhishek Parakh, Subhash Kak "Online data storage using implicit security", Elsevier science Direct on Information Sciences, 2009.

[10] Dimitrios Zissis, Dimitrios Lekkas, "Addressing cloud computing security issues", Elsevier science Direct on Future Generation Computer Systems, 2012.

[11] Giuliano Garrammone, and Balazs Matuz, "Short Erasure Correcting LDPC IRA Codes over GF(*q*)", IEEE Global Telecommunications Conference, 2010.

[12] Yan Zhu, Hongxin Hu, Gail-Joon Ahn, Mengyang Yu, "Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage", IEEE Transactions on Parallel and Distributed Systems, 2012.

[13] Ms. Disha H. Parekh, Dr. R. Sridaran, "An Analysis of Security Challenges in Cloud Computing", International Journal of Advanced Computer Science and Applications, Vol. 4, No.1, 2013.

[14] Saurabh Kumar Garg, Steve Versteeg, Rajkumar Buyya, "A framework for ranking of cloud computing services", Elsevier science Direct on Future Generation Computer Systems, 2013.

[15] Satyakshma Rawat, Richa Chowdhary, and Dr. Abhay Bansal, "Data Integrity of Cloud Data Storages (CDSs) in Cloud", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013.

First Author B Durairaj Moorthy is currently pursuing a PhD at the Karpagam University in India. He holds a BSc in Computer Technology and an MCA. He has over 8 years of experience in Network Management system and implementation of large and complex network management in cloud environment. He perceived best outgoing student award in both UG and PG courses and he is a topper of class in MCA. Secured Star performance, PANORAMA, Productivity and Quality awards for various deliveries in the projects he involved. Provided various value adds in his product and implemented the same as well. His research interest lies in store the data in cloud in secure way and effective retrieval.

Second Author Dr. C. Chandrasekar is an Associate Professor at Department of Computer Science - Periyar University, Salem, India. He perceived MCA and Ph.D and trained 16 plus students for their research work. He is having 16 plus years of experience in teaching and research. He was did research in Mobile and Wireless Computing.