Volume 2, Issue 12, December 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Extreme Confidential Component Mining Based on Coupling and Cohesion Strategy in Cloud Computing

Iyapparaja. M* *Asst. Professor-CSE, K.S.R. College of Engineering, Tiruchengode, Tamilnadu.*

Dr. Sureshkumar. S *Principal, Vivekanandha College of Technology for Women , Tiruchengode, Tamilnadu.*

Anand. R *Final year M.E, K.S.R. College of Engineering Tiruchengode, Tamilnadu*

Abstract— ***Cloud computing provides computation environment for the software, data access, and storage resources without requiring cloud users to know the location and other details of the computing infrastructure. It delivers convenient, on-demand access to shared pools of data, applications and hardware over the internet. Cloud computing provides unlimited infrastructure to store and execute customer data and program. In this paper auditing ensures the data correctness guaranteed through the symmetric keys from the three different server space. This paper also achieves the concept of Coupling and Cohesion for the shared data in the cloud. The proposed system also supports for the dynamic data operation such as append, insert, and delete. It also supports in identifying the misbehaving servers through distributed erasure-coded data. It also adapt to the situations like Byzantine failure, malicious data modification attack, and even server colluding attacks.***

Keywords— cloud computing, Coupling, Cohesion, symmetric keys, data dynamics.

I. INTRODUCTION

Cloud computing is the delivery of computing as a service rather than a product, where by shared resources, software and information are provided to computers and other devices as a utility (like the electricity grid) over a network (typically the Internet). Cloud computing provides computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Parallels to this concept can be drawn with the electricity grid, where in end-users consume power without needing to understand the component devices or infrastructure required to provide the service. As customers we do not need to own the infrastructure, they are merely accessing or renting; they can forego capital expenditure and consume resources as a service, paying instead for what they use. Data can be redundantly store in multiple physical locations. Due to this redundancy the data can be easily modified by unauthorized users which can be stored in the database. This leads to loss of data privacy and security to database. Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users now subscribe high quality services from data and software that reside solely on remote data centers.

Extensive and performance analysis shows that the proposed scheme ensures that cyclic redundancy check and time-tested practices and technologies for managing trust relationships in traditional enterprise IT environments can be extended to work effectively in both private and public clouds. Those practices include data encryption, strong Authentication and fraud detection. Coupling and cohesion measures capture the degree of interaction and relationships among source code elements, such as classes, methods, and attributes in object-oriented (OO) software systems. One of the main goals behind OO analysis and design is to implement a software system where classes have high cohesion and low coupling among them. These class properties facilitate comprehension activities, testing efforts, reuse, and maintenance tasks. A vast majority of Coupling and Cohesion metrics abound in the literature relies on structural information, which captures relations, such as method calls or attributes usages. These metrics have been proved useful in different tasks, such as, assessment of design quality, impact analysis, prediction of software quality, and faults, Identification of design patterns etc. However, these structural metrics lack the ability to identify conceptual links, which, for example, specify implicit relationships encoded in identifiers and comments in source code.

II. PROBLEM STATEMENTS

The following subsequent of steps have been carried out during system model

A. User

The user has the data to store in the cloud the cloud area may be the enterprise or organizational based cloud or individual cloud [1][2].

B. Cloud service provider

It is the storage space from the cloud server to store the data in the cloud area owns and operates on the cloud computing system [1][2].

C. Third Party Auditor (TAP)

The third party auditor has the capability and expert in finding the risk in the cloud computing and to made the storage in the cloud efficient. [1][3][9].

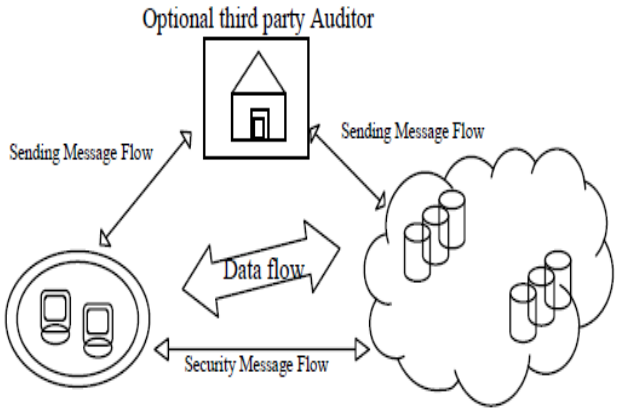

Figure: 1 Architecture of cloud computing

As users no longer possess their data locally, it is of critical importance to assure users that their data are being correctly stored and maintained. That is, users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. n case those users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted TPA of their respective choices. We assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead.

Recently the importance of ensuring the remote data integrity has been highlighted by the some research works. These techniques, while can be useful to ensure the storage correctness without having users possessing data, cannot address all the security threats in cloud data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations.

As a complementary approach, researchers have also proposed distributed protocols for ensuring storage correctness across multiple servers to peers. Again, none of these distributed schemes is aware of dynamic data operations. As a result, their applicability in cloud data storage can be drastically limited. Compared to many of its predecessors, which only provide binary results about the storage state across the distributed servers, the challenge-responses protocol further provides the localization of data error[4].

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. To ensure the security and dependability for cloud data storage under the aforementioned adversary model, we aim to design efficient mechanisms for dynamic data verification and operation and achieve the following goals.

D Storage Correctness To ensure the data are kept intact all the time in the cloud [5].

E Fast Localization Of Data Error To effectively locate the malfunctioning server when data corruption has been detected.[5].

F Dynamic Data Support To maintain the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud.[5].

G Dependability To enhance data availability against Byzantine failures, malicious data modification and server colluding attacks.[5].

H Light weight To enable users to perform storage correctness checks with minimum overhead.[6]

In this project, an effective and flexible distributed scheme is proposed with explicit dynamic data support to ensure the correctness of users' data in the cloud. Some part of the data is encrypted in the cloud storage with a symmetric key encryption. For example, the day to day transaction database records. Their aggregated values are encrypted in data owner storage which is less in size. The database in cloud storage may be redundant and can be accessed by more number of users.

To check the data in cloud storage is safe, the sample data can be fetched from cloud storage and decrypted [5]. The aggregated data is also decrypted so that the data from cloud produce the same aggregated data. This ensures the data in the cloud storage is unaffected by users. The storage correctness verification is made in the above manner. In addition, each and every user is provided with a strong authentication mechanism so that the data owner, user, cloud provider is

able to access with more trusted mechanism. A policy is induced and checked periodically to check the data is not violated.[5][6].

III. COHESION & COUPLING METRICS

The definitions of the new conceptual cohesion and coupling of classes build on our previous work for measuring the conceptual cohesion and coupling of classes. The source code of the software system is parsed and transformed into a corpus of textual components where each component corresponds to the implementation of a method. Aforementioned LSI technique takes the corpus as an input and creates a term-by-component matrix, which captures the dispersion and co-occurrence of terms in class methods. Singular value
Decomposition (SVD) is used next to construct a subspace, referred to as the Latent Semantic Indexing (LSI) subspace.
All methods from this matrix are represented as vectors in the LSI subspace. The cosine similarity between two vectors is used as a measure of conceptual similarity between two methods and is purported to determine shared conceptual information between two methods in the context of the entire software system[6].
This mechanism to capture conceptual similarity among components has been introduced before in Conceptual Coupling of Classes and Conceptual Cohesion of Classes measures and is also used here. we also include them for the sake of completeness.[16]

IV. ENSURING CLOUD DATA STORAGE

In cloud data storage system, users store their data in the cloud and no longer possess the data locally. Thus, the correctness and availability of the data files being stored on the distributed cloud servers must be guaranteed [6]. One of the key issues is to effectively detect any unauthorized data modification and corruption, possibly due to server compromise and/or random Byzantine failures. Besides, in the distributed case when such inconsistencies are successfully detected, to find which server the data error lies in is also of great significance, since it can be the first step to fast recover the storage errors. To address these problems, our main scheme for ensuring cloud data storage is presented in this section.[15]
The first part of the section is devoted to a review of basic tools from coding theory that is needed in our scheme for file distribution across cloud servers. Then, the symmetric key is introduced. The token computation function we are considering belongs to a family of universal hash function [15], chosen to preserve the symmetric properties, which can be perfectly integrated with the verification of erasure-coded data[7][16]. Subsequently, it is also shown how to derive a challenge response protocol for verifying the storage correctness as well as identifying misbehaving servers. Finally, the procedure for file retrieval and error recovery based on erasure-correcting code is outlined.

V. FILE DISTRIBUTION PREPARATION

It is well known that erasure-correcting code may be used to tolerate multiple failures in distributed storage systems. In cloud data storage, we rely on this technique to disperse the data file F redundantly across a set of n = m+ k distributed servers. A (m + k, k) Reed-Solomon erasure-correcting code is used to create k redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the m + k data and parity vectors. By placing each of the m + k vectors on a different server, the original data file can survive the failure of any k of the m+k servers without any data loss, with a space overhead of k/m. For support of efficient sequential I/O to the original file, our file layout is systematic, i.e., the unmodified m data file vectors together with k parity vectors is distributed across m+ k different servers.[1][7].

Let **F** = (F1, F2, . . . , Fm) and Fi = (f1i, f2i, . . . , fli)T (i ∈ {1, . . . ,m}), where l ≤ 2p − 1. Note all these blocks are elements of GF(2p). The systematic layout with parity vectors is achieved with the information dispersal matrix A, derived from an m × (m+ k) Vander mode matrix.

$$\begin{bmatrix} 1 & 1 & \ldots\ldots\ldots\ldots 1 & 1 & 1 \\ \beta 1 & \beta 1 & \ldots\ldots\ldots\ldots \beta m & \beta m+1 & \beta n \\ . & . & \ldots\ldots\ldots\ldots & . & . \end{bmatrix}$$

where βj (j ∈ {1, . . . , n}) are distinct elements randomly picked from GF(2p). After a sequence of elementary row transformations, the desired matrix A can be written as given below matrix [1][7].

$$A = (I|P) = \begin{bmatrix} 1 & 0 & 0 & p11 & p12 \ldots . P1k \\ 0 & 1 & 0 & p21 & p22 \ldots p2k \\ . & . & . & . & . & . \\ 0 & 0 & 1 & pm1 & pm2 & pmk \end{bmatrix}$$

where I is a m×m identity matrix and P is the secret parity generation matrix with size m × k. Note that A is derived from a Vander monde matrix, thus it has the property that any m out of the m+ k columns form an invertible matrix.[12][13].

By multiplying **F** by **A**, the user obtains the encoded file:

$$G = F \cdot A = (G(1),G(2), \ldots ,G(m),G(m+1), \ldots ,G(n)) = (F1, F2, \ldots , Fm,G(m+1), \ldots ,G(n)),$$

where $G(j) = (g(j)\ 1\ , g(j)\ 2\ , \ldots , g(j)\ l\ )T\ (j \in \{1, \ldots , n\})$.

As noticed, the multiplication reproduces the original data file vectors of **F** and the remaining part (G(m+1), . . . ,G(n)) are k parity vectors generated based on **F**.[1][7]

Whenever the data corruption is detected, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server(s) (again with high probability).

Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by erasure correction, , as long as the number of identified misbehaving servers is less than k. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.[11][12].

VI. TOKEN AND SYMMETRIC KEY PRECOMPUTATION

In order to make sure the data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens and the symmetric key. Before file distribution the user pre-computes a certain number of short verification tokens on individual vector each token covering a random subset of data blocks [13].

Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user.

The values of these signatures should match the corresponding tokens pre-computed by the user. Mean while, as all servers operate over the same subset of the indices, the requested response values for integrity check must also be a valid codeword determined by secret matrix P. Together with the token symmetric key will be used to make sure the user verification with the cloud data retrieval. The symmetric keys from the stored locations of the different cloud matches with each other authentication to the user will be provided [13].

VII. CLOUD DATA CORRECTNESS VERIFICATION AND LOCALIZATION OF ERRORS

Error localization is a key prerequisite for eliminating errors in storage systems. It is also of critical importance to identify potential threats from external attacks. However, many previous schemes [2], do not explicitly consider the problem of data error localization, thus only providing binary results for the storage verification. Our scheme outperforms those by integrating the correctness verification and error localization (misbehaving server identification) in our challenge-response protocol: the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error(s)[12].

Once the inconsistency among the storage has been successfully detected, we can rely on the precomputed verification tokens to further determine where the potential data error(s) lies in. It is computed exactly in the same way as token, thus the user can simply find which server is misbehaving by verifying the correctness verification algorithm[2][8].

VIII. RETRIEVING THE FILE AND ERROR RECOVERY

Since our layout of file matrix is systematic, the user can reconstruct the original file by downloading the data vectors from the first m servers, assuming that they return the correct response values. Notice that our verification scheme is based on random spot-checking, so the storage correctness assurance is a probabilistic one. However, by choosing system parameters appropriately and conducting enough times of verification, we can guarantee the successful file retrieval with high probability.

On the other hand, whenever the data corruption is detected, the comparison of precomputed tokens and received response values can guarantee the identification of misbehaving server(s) (again with high probability), which will be discussed shortly[14]. Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by erasure correction, shown in error recovery, as long as the number of identified misbehaving servers is less than k. (otherwise, there is no way to recover the corrupted blocks due to lack of redundancy, even if we know the position of misbehaving servers.) The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage [2][10].

IX. TOWARD THIRD PARTY AUDITING

As discussed in our architecture, in case the user does not have the time, feasibility, or resources to perform the storage correctness verification, he can optionally delegate this task to an independent third-party auditor, making the cloud storage publicly verifiable. However, as pointed out by the recent work to securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities toward user data privacy. Namely, TPA should not learn user's data content through the delegated data auditing. Now we show that with only slight modification, our protocol can

support privacy-preserving third party auditing. The new design is based on the observation of linear property of the parity vector blinding process. Recall that the reason of blinding process is for protection of the secret matrix P against cloud servers. However, this can be achieved either by blinding the parity vector or by blinding the data vector (we assume k < m). Thus, if we blind data vector before file distribution encoding, then the storage verification task can be successfully delegated to third party auditing in a privacy-preserving manner[3][9].

X. DYNAMIC OPERATION SUPPORTS

A Update Operation

In cloud data storage, a user may need to modify some data block(s) stored in the cloud, from its current value to a new one. We refer this operation as data update. The high level logical representation of data blocks update. Due to the linear property of Reed-Solomon code, a user can perform the update operation and generate the updated parity blocks by using specific matrix, without involving any other unchanged blocks[7][8][11].

B Delete Operation

Sometimes, after being stored in the cloud, certain data blocks may need to be deleted. The delete operation we are considering is a general one, in which user replaces the data block with zero or some special reserved data symbol. From this point of view, the delete operation is actually a special case of the data update operation, where the original data blocks can be replaced with zeros or some predetermined special blocks. Therefore, we can rely on the update procedure to support delete operation, i.e., by setting. Also, all the affected tokens have to be modified and the updated parity information has to be blinded using the same method specified in an update operation.[7][8][9].

C Append Operation

In some cases, the user may want to increase the size of his stored data by adding blocks at the end of the data file, which we refer as data append. We anticipate that the most frequent append operation in cloud data storage is bulk append, in which the user needs to upload a large number of blocks (not a single block) at one time[16].

Given the file matrix F illustrated in file distribution Preparation, appending blocks toward the end of a data file is equivalent to concatenate corresponding rows at the bottom of the matrix layout for file F . In the beginning, there are only l rows in the file matrix. To simplify the presentation, we suppose the user wants to append m blocks at the end of file F, (We can always use zero-padding to make a row of m elements). With the secret matrix P, the user can directly calculate the append blocks for each parity server. To ensure the newly appended blocks are covered by our challenge tokens, we need a slight modification to our token precomputation.[7][8].

D Insert Operation

An insert operation to the data file refers to an append operation at the desired index position while maintaining the same data block structure for the whole data file, i.e., inserting a block j_ corresponds to shifting all blocks starting with index j by one slot.

Thus, an insert operation may affect many rows in the logical data file matrix F, and a substantial number of computations are required to renumber all the subsequent blocks as well as recomputed the challenge-response tokens. Hence, a direct insert operation is difficult to support. In order to fully support block insertion operation, recent work [7][14][15], suggests utilizing additional data structure (for example, Merkle Hash Tree) to maintain and enforce the block index information. Following this line of research, we can circumvent the dilemma of our block insertion by viewing each insertion as a logical append operation at the end of file F. Specifically, if we also use additional data structure to maintain such logical to physical block index mapping information, then all block insertion can be treated via logical append operation, which can be efficiently supported. On the other hand, using the block index mapping information, the user can still access or retrieve the file as it is. Note that as a trade off, the extra data structure information has to be maintained locally on the user side.[7][8][15].

XI. CONCLUSION

This paper, investigate cloud data storage data security problem, which is 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, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By using the homomorphic token with distributed verification of erasure coded data with the symmetric key encryption and decryption technique, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s).

Considering the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third party auditors and be worry-free to use the cloud storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

REFERENCES

[1] Cong Wang, Qian Wang, , Kui Ren, Senior Member, "*Toward Secure and Dependable Storage Services in Cloud Computing*" IEEE, Ning Cao, and Wenjing Lou, Senior Member IEEE transactions on services computing, vol. 5, no. 2, April-June 2012

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

[3] 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.

[4] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik,"*Scalable and Efficient Provable Data Possession"* .Fourth Int'l Conf. Security and Privacy in Comm. Netowrks (SecureComm '08), pp. 1-10, 2008.

[5] 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.

[6] Cong Wang, Qian Wang, and Kui Ren Department of ECE, Illinois Institute of Technology, "Ensuring *Data Storage Security in Cloud Computing*"June2009.

[7] Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, Ram Swaminathan (HP Labs, Palo Alto) "*Auditing to Keep Online Storage Services Honest* "April 2010.

[8] Mehul A. Shah, Ram Swaminathan, Mary Baker, HP Laboratories, Palo Alto, *"Privacy-Preserving Audit and Extraction of Digital Contents"* HPL-2008-32R1, April 30, 2009.

[9] White Paper November 2009 2 Building Customer Trust in Cloud Computing with Transparent Security Sun Microsystems, Inc. "*Building Customer Trust In Cloud Computing With Transparent Security*"2010.

[10] Ari Juels and Burton S. Kaliski Jr RSA Laboratories Bedford, MA, USA, "*PORs: Proofs of Retrievability for Large Files*" ajuels@rsa.com EMC Corporation Hopkinton, MA, USA kaliski

[11] Giuseppe Ateniese Randal Burns Reza Curtmola Joseph Herring Lea Kissner Zachary Peterson Dawn Song, "*Provable Data Possession at Untrusted Stores*".

[12] Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik, "*Scalable and Efficient Provable Data Possession*".

[13] Qian Wang, Cong Wang, Jin Li1, Kui Ren, and Wenjing Lou Illinois Institute of Technology, Chicago IL 60616, USA, "*Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud computing.*

[14] L. Carter and M. Wegman, "*Universal Hash Functions*," Journal of Computer and System Sciences, vol. 18, no. 2, pp. 143–154, 2009.

[15] T . S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," *ICDCS '06*, pp. 12–12, 2006.

[16] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E., "*Recovering Traceability Links between Code and Documentation*", *IEEE Transactions on Soft. Engineering*, vol. 28, no. 10, pp. 970 - 983., 2002

AUTHORS PROFILE

M. Iyapparaja is presently a Senior Assistant Professor, Department of CSE, K.S.R. College of Engineering, Tiruchengode, Namakkal, Tamilnadu, India. He received the B.E degree from Anna University, Chennai and M.E degree from Anna University of Technology, Coimbatore in 2006 and 2010 respectively. His research interests include Software Testing, Software Engineering and Agile Testing. He is a member of ISTE.

Dr. **S. Sureshkumar** is presently the Principal, Vivekanandha College of Technology for Women, Tiruchengode. He received the B.E degree from National Engineering College, M.S, degree in Software system from Birla Institute of Technology and Science, M.Tech., degree from Indian Institute of Technology, Kharagpur and Ph.D., degree from Anna University, Chennai in 1988, 1993, 2000 and 2009 respectively. He has published 30 numbers of papers in refereed international journals and conferences. His research interests include Image processing, parallel processing and Energy. He is a member of ISTE and IET.

