International Journal of Engineering Innovation & Research
Volume 2, Issue 6, ISSN: 2277 – 5668

Adversary Model Operation in Cloud Computing

A. Balakrishna, P. Suresh Babu, S. Nageswara Rao, T. Murali Mohan
Department of CSE, Kaushik College of Engineering, Visakhapatnam, Andhra Pradesh, India

***Abstract* – As cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud. Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest, these techniques support only exact keyword find out. In this paper, for the first time we formalize and solve the problem of effective fuzzy keyword find out over encrypted cloud data while maintaining keyword privacy. Fuzzy keyword search greatly enhances system usability by returning the matching files when users searching inputs exactly match the predefined keywords or the closet possible matching files based on keyword similarity semantics, when exact match fails. In our solution, we exploit the distance to quantify keywords similarity and develop two advanced techniques on constructing fuzzy keywords sets, which achieve optimized storage an representation overheads. We further proposed a brand new symbol-based trie-traverse searching scheme, where a multi-way tree structure is built up using symbols transformed from the resulted fuzzy keyword sets. Through rigorous security analysis, we show that our proposed solution is secure and privacy preserving, while correctly realizing the goal of fuzzy keyword search. Extensive experimental results demonstrate the efficiency of the proposed solution.**

***Keywords* – Cloud Computing, Semantics, Symbol-Based.**

I. Introduction

Number of trends are opening up the era of cloud computing, which is an internet based development and use of computer technology. As the number of users are increasing day by day increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscriber high quality services from data and software that reside solely on remote data centers [1,2]. When moving data in the cloud offers great convenience to the users since they don't have to care about the complexity of direct hardware management. From the perspective of data security, which has always been an important aspect of quality of service cloud computing inevitable poses new challenges security, which has been always been an important aspect of quality of service cloud computing inevitably poses new challenges security threats or number of reason. Firstly, traditional cryptography primitive. Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works [3,4,5]. These techniques, which can be useful to ensure the storage, since they are focusing on single server scenario and most of them do not consider protocols for ensuring storage correctness across multiple servers or peers[6,7,8,9,10].

In this paper, we propose an effective and flexible distributed scheme with explicitly dynamic data support to ensure the correctness of users data in cloud. We rely on erasure corrective codes in their redundancy and guarantee the data dependability this construction drastically reduces the communication and storage overhead as compared to the traditional based file our work is among the first new one in this filed to consider distributed data storage in cloud computing.

II. Proposed System

Modules:
1. System Models
2. Adversary Models
3. Design Goals

Problem Statement:

1. System Models:

In system models we discuss three different networks can be identified as follows:

- User: Users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
- Could Service Provider(CSP): A CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live cloud computing systems.
- Third Party Auditor (TPA): An optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

In cloud Computing data storage, whenever the user wants to stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, we are co-operated and distributed manner,. Data redundancy ca n be employed with techniques of erasure correcting code to further tolerate faults/ servers. When there are the application purpose, the user interacts with the cloud servers through CSP to access or retrieve the data. In most cases these operations are to be considering i.e., Update, delete, insert and append.

2. Adversary Model:

Security threats are faced by cloud data can be different sources. On the one hand, a CSP can be self interested, and not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may labels due to management errors. And on the other hand, there must be also exist an economically-motivated adversary, who has the capability to compromises a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete users data while remaining undetected by CSPs for a certain period of time.

Weak Adversary: When interesting a corrupting the user's data files stored on individual servers. Once a server is comprised, an adversary can pollute the original data

International Journal of Engineering Innovation & Research
Volume 2, Issue 6, ISSN: 2277 – 5668

files by modifying or introducing their own fraudulent data to prevent the original data from being retrieved by the user.

Strong Adversary: when Worst case scenario, we assume that the adversary can compromise all the storage servers so that he can internationally modify the data file as long as they are internally consistent. In fact, this is equivalent to the case where all servers are colluding together to hide a data loss.

3. Design Goals:

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: (1) Storage correctness: to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. (2) Fast localization of data error: to effectively locate the mal-functioning server when data corruption has been detected. (3) 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. 4) Dependability: to enhance data availability against Byzantine failures, malicious data modification and server colluding attacks, i.e. minimizing the effect brought by data errors or server failures. (5) Lightweight: to enable users to perform storage correctness checks with minimum overhead.

II. ENSURING CLOUD DATA STORAGE

In cloud storage systems, 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. To address these problems, our main scheme for ensuring cloud data storage is presented in this section. The first part of the section represented to review a basic tool from coding theory that are needed in our scheme for file distribution across cloud servers.

A. 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.

Let $\mathbf{F} = (F_1, F_2, \ldots, F_m)$ and $F_i = (f_{1i}, f_{2i}, \ldots, f_{li})^T$ (i ▫ {1, . . . , m}), where $l \le 2^p - 1$. Note all these blocks are elements of GF (2^p).

After a sequence of elementary row transformations, the desired matrix A can be written as

$$A \quad I P$$

$$\begin{matrix} 1 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1k} \\ 0 & 1 & \cdots & 0 & p_{21} & p_{22} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{m1} & p_{m2} & \cdots & p_{mk} \end{matrix}$$

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.

Algorithm 1: Token Pre-computation

1: **procedure**
2: Choose parameters l, n and function f, φ;
3: Choose the number t of tokens;
4: Choose the number r of indices per verification;
5: Generate master key K_{prp} and challenge k_{chal};
6: **for** vector $G^{(j)}$, j ← 1, n **do**
7: **for** round i ← 1, t **do**
10: **end for**
11: **end for**
12: Store all the v_is locally.
13: **end procedure**

B. Challenge Token Precomputation

In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main ide a is as follows: before file distribution the user pre-compute s a certain number of short verification tokens on individual vector $G^{(j)}$ (j ▫ {1, . . . , n}), each token covering a random subset of data blocks. 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 the m to the user. The values of these signatures should match the corresponding tokens pre-computed by the user. Meanwhile, 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.

Algorithm 2:

Correctness Verification and Error Localization

1. procedure CHALLENGE(i)
2. Recompute $\alpha_i = f_{kchal}$ (i) and $k_{prp}^{(i)}$ from $K_{P\ 3.RP}$ Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
4.Receive from servers: $\{R^{(j)\ Pr}\ \alpha^q \quad G^{(j)}(q)] | 1 \le j \le n\}_{\ i\ q=1}$
$_i$ [φ_k(i)= prp
5:for (j ← m + 1, n) do
6: $R^{(j)} \leftarrow R^{(j)} - {}^{Pr}_{q=1}\ f_{kj}\ (s_{Iq\ ,j}) \cdot \alpha^q_i$, $I_q = \varphi_k$(i) (q) prp end for

if ($(R_i^{(1)}, \ldots, R_i^{(m)}) \cdot P == (R_i^{(m+1)}, \ldots, R_i^{(n)})$) then Accept and ready for the next challenge.
else
for (j ← 1, n) do
if ($R_i^{(j)} != v_i^{(j)}$) then
13: return server j is misbehaving.
end if
end for
end if
end procedure

users stores them locally to obviate the need for encryption and lower the bandwidth overhead during dynamic data operation which will be discussed shortly. where k_j is the secret key for parity vector $G^{(j)}$ ($j \in \{m + 1, \ldots, n\}$). This is for protection of the secret matrix P. We will discuss the necessity of using blinded parities in detail in Section V. After blinding the parity information, the user disperses all the n encoded vectors $G^{(j)}$ ($j \in \{1, \ldots, n\}$) across the cloud servers $S_1, S_2, \ldots, S_n$.

C. File Retrieval and Error Recovery

Since our layout of file matrix is systematic, the user can reconstruct the original file by downloading the data vector s from the first m servers, assuming that they return the correct response values. Notice that our verification scheme is base d on random spot-checking, so the storage correctness assurance is a probabilistic one. However, by choosing system parameters (e.g., r, l, t) 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 pre-computed tokens and received response values can guarantee the identification of misbehaving server(s), again with high probability, which will be discussed shortly. 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 Algorithm 3, as long as there are at most k misbehaving servers are identified. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

III. Security Analysis and Performance Evaluation

A. Security Strength against Weak Adversary

1) Detection Probability against data modification: In our scheme, servers are required to operate on specified rows in each correctness verification for the calculation of requested. We first analyze the matching probability that at least one of the rows picked by the user matches one of the rows modified by the adversary:

$$P_m^r = 1 - P\{X = 0\}$$
$$= 1 - \left(1 - \min\left\{\frac{z}{\cdot}, 1\right\}\right)^{r-1}$$
$$= 1 - \left(\frac{l - z}{l}\right)^{r}.$$

B. Security Strength Against Strong Adversary

In this section, we analyze the security strength of our schemes against server colluding attack. Recall that in the file distribution preparation, the redundancy parity vectors are calculated via multiplying the file matrix **F** by **P**, where **P** is the secret parity generation matrix we later rely on for storage correctness assurance. If we disperse all the generated vectors directly after token precomputation, i.e., without blinding, malicious servers that collaborate can reconstruct the secret P matrix easily: they can pick blocks from the same rows among the data and parity vectors to establish a set of m · k linear equations and solve for the m . k entries of the parity generation matrix **P**. Once they have the knowledge of **P**, those malicious servers can consequently modify any part of the data blocks and calculate the corresponding parity blocks, and vice versa, making their codeword relationship always consistent. Therefore, our stor-age correctness challenge scheme would be undermined even if those modified blocks are covered by the specified rows, the storage correctness check equation would always hold.

C. Performance Evaluation

1) File Distribution Preparation: We implemented the gen-eration of parity vectors for our scheme under field GF (2^8). Our experiment is conducted using C on a system with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM, and a 7200 RPM Western Digital 250 GB Serial ATA drive with an 8 MB buffer. We consider two sets of different parameters for the (m + k, m) Reed-Solomon encoding. Table I shows the average encoding cost over 10 trials for an 8 GB file. In the table on the top, we set the number of parity vectors constant at 2. In the one at the bottom, we keep the number of the data vectors fixed at 8, and increase the number of parity vectors.

Table I: The cost of parity generation in seconds for an 8GB

set I	m = 4	m = 6	m = 8	m = 10
k = 2	567.45s	484.55s	437.22s	414.22s

set II	k = 1	k = 2	k = 3	k = 4
m = 8	358.90s	437.22s	584.55s	733.34s

IV. Related Work

Whenever we combines spot checking and error correcting codes to ensure both possession and irretrievability of files on archive service systems. In this model and constructed a random linear function based homomorphic authenticator which enables unlimited number of queries and requires less communication overhead. However, all these schemes are focusing on static data. The effectiveness of their schemes rests primarily the preprocessing steps that the user conducts before outsourcing the data file F. Any change to the

contents of F, even few bits, must propagate through the error correcting code, thus introducing significant computation and communication complexity. This method has lower-overhead than their previous scheme and allows for block update, deletion and append to the stored file, which has also been supported in our work, However, their scheme focuses on single server scenario and does not address small data corruptions, leaving both the distributed scenario and data error recovery issues unexplored.

In other related work, backup schemes in which blocks of the data files are dispersed across m+k peers using an erasure code. Peers can be request random blocks from their backup peers and verify the integrity using separate keyed cryptographic hashes attached on each block. However, their proposal requires exponentiation over the entire data file, which is clearly impractical for the server whenever the files is large.

V. CONCLUSION

In this paper, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append.. By utilizing the homomorphism token with distributed verification of erasure - coded data, 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 servers. We believe that the data storage security in cloud computing, an area full of challenge and paramount importance, is still infancy now, and many research problems are rest to be identified.

REFERENCES

[1] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrie vability for Large Files," *Proc. of CCS '07*, pp. 584–597, 2007.

[2] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievab ility: Theory and Implementation," Cryptology ePrint Archive, Report 20 08/175, 2008, http://eprint.iacr.org/.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores, " *Proc. of CCS '07*, pp. 598–609, 2007.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," *Proc. of SecureComm '08*, pp. 1– 10, 2008.

[5] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," *Proc. of the 2003 USENIX Annual Technical Conference (General Track)*, pp. 29–41, 2003.

[6] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Avail ability and Integrity Layer for Cloud Storage," Cryptology ePrint Arch ive, Report 2008/489, 2008, http://eprint.iacr.org/.

[7] L. Carter and M. Wegman, "Universal Hash Functions," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 143–154, 1979.

[8] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Dist ributed Erasure-coded Data," *Proc. 26th ACM Symposium on Principles of Distributed Computing*, pp. 139–146, 2007.

[9] J. S. Plank and Y. Ding, "Note: Correction to the 1997 Tut orial on Reed-Solomon Coding," University of Tennessee, Tech. Rep. CS-03-504, 2003.

[10] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Sec ure Sensor Data Storage with Dynamic Integrity Assurance," *Proc. of IEEE INFOCOM*, 2009.

[11] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP : Multiple-Replica Provable Data Possession," *Proc. of ICDCS '08*, pp. 411–420, 2008.

[12] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Au diting to Keep Online Storage Services Honest," *Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07)*, pp. 1–6, 2007.

AUTHOR'S PROFILE

A. Bala Krishna
Student of Kaushik College of Engineering,
M. Tech.,CSE, 2nd Year,
Visakhapatnam, Andhra Pradesh.

Sri. P. Suresh Babu,
B. Tech., M.E., CSI, Associate Professor,
Department of Computer Science & Engineering
Kaushik College of Engineering,
Visakhapatnam, Andhra Pradesh.
Teaching Experience: 14 Years
Industrial Experience: 4 Years

S. Nageswara Rao,
M. Sc, Ph. D
Department of Computer Science & Engineering
Student of Kaushik College of Engineering,
Visakhapatnam, Andhra Pradesh.

T. Murali Mohan
M.Tech. (CSE), MBA, Asst., Professor
Department of Computer Science & Engineering
Kaushik College of Engineering, Visakhapatnam, Andhra Pradesh.
Teaching Experience: 4Years