ISSN:0975-9646

CH.Venkata lakshmi et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (6), 2012,5374-5377

Secure and Dependable Storage Services In Cloud Computing

CH.Venkata lakshmi, Jammi Ashok
Dept. of CSE, Guru Nanak Institute of Technology ,
Hyderabad

Abstract- Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. Though the benefits are clear, such a service is also relinquishing users 'physical possession of their outsourced data, which inevitably poses new security risks towards the correctness of the data in cloud. In order to address this new problem and further achieve a secure and dependable cloud storage service, I propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphism token and distributed erasure-coded data.

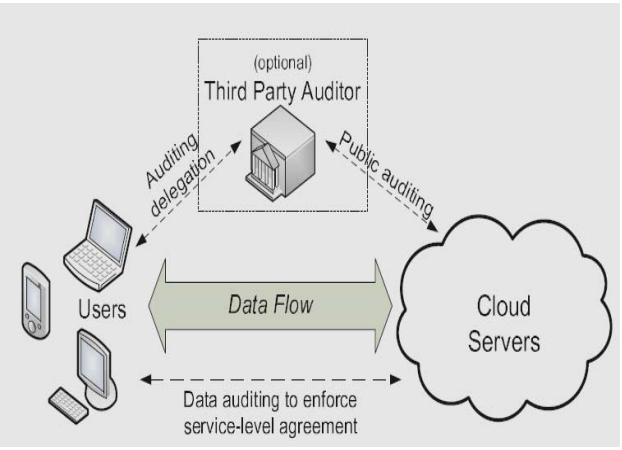
The proposed design allows users to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the Identification of misbehaving server. Considering the cloud data are dynamic in nature, the proposed design further supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

1. INTRODUCTION

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 can now subscribe high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management.

A representative network architecture for cloud storage service architecture is illustrated in Figur1

.



Three different network entities can be identified as follows:

• **User**: an entity, who has data to be stored in the cloud and relies on the cloud for data storage and computation, can be either enterprise or individual customers.

• **Cloud Server (CS)**: an entity, which is managed by *cloud service provider* (CSP) to provide data storage service and has significant storage space and computation resources • **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 data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy can be employed with technique of erasure correcting code to further tolerate faults or server crash as user's data grows in size and importance. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data .In some cases, the user may need to perform block level operations on his data.

1.1 EXISTING SYSTEM

In existing system, the importance of ensuring the remote data integrity has been highlighted by the following research works under different system and security models. These techniques, while can be useful to ensure the storage correctness without having users possessing local data, are all focusing on single server scenario. They may be useful for quality-of-service testing, but does not guarantee the data availability in case of server failures. Although direct applying these techniques to distributed storage (multiple servers) could be straightforward, the resulted storage verification overhead would be linear to the number of servers.

1.2 PROBLEMS IN EXISTING SYSTEM

- 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 capability of handling dynamic data remains unclear, which inevitably limits their full applicability in Server storage scenarios.

2. RELATED WORK

Juels et al. described a formal "proof of retrievability" (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error correcting code to ensure both possession and retrievability of files on archive service systems. Shacham et al. built on this model and constructed a random linear function based homomorphic authenticator which enables unlimited

number of challenges and requires less communication overhead due to its usage of relatively small size of BLS signature. Ateniese *et al.* defined the "provable data possession" (PDP) model for ensuring possession of file on untrusted storages. Their scheme utilized public key based homomorphic tags for auditing the data file. However, the pre-computation of the tags imposes heavy computation overhead that can be expensive for an entire file. In their subsequent work, Ateniese *et al.* described a PDP scheme that uses only symmetric key based cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not provide data availability guarantee against server failures, leaving both the distributed scenario and data error recovery issue unexplored. The incremental cryptography work done by Bellare *et al.* also provides a set of cryptographic building blocks such as hash, MAC, and signature functions that may be employed for storage integrity verification while supporting dynamic operations on data. Schwarz *et al.* proposed to ensure static file integrity across multiple distributed servers, using erasure-coding and block level file integrity checks. We adopted some ideas of their distributed storage verification protocol. However, our scheme further support data dynamics and explicitly study the problem of misbehaving server identification

3.0 METHODS

3.1 Challenge Token Pre-computation:

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 idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector G(j) (j $\in$ {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 them 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 1 Token Pre-computation

1: **procedure**
2: Choose parameters l, n and function f, Ø;
3: Choose the number t of tokens;
4: Choose the number r of indices per verification;
5: Generate master key K_{PRP} and challenge key k_{chal};
6: **for** vector G(j), j ← 1, n **do**
7: **for** round i← 1, t **do**
8: Derive $\alpha_i = fk_{chal}(i)$ and $k^{(i)}_{prp}$ from K_{PRP}.
9: Compute $v^{(j)} = \Sigma^{r}_{q=1} \alpha^{q}_{i} * G^{(j)}[Ø\ k^{(i)}_{prp}(q)]$
10: **end for**
11: **end for**
12: Store all the v_i's locally.
13: **end procedure**

Suppose the user wants to challenge the cloud servers t times to ensure the correctness of data storage. Then, he must pre-compute t verification tokens for each G(j) (j$\in${1, . . . , n}), using a PRF f(·), a PRP _(·), a challenge key k_{chal} and a master permutation key K_{PRP} . Specifically, to generate the i^{th} token for server j, the user acts as follows:

1) Derive a random challenge value α_i of GF(2^p) by $\alpha_i = fk_{chal}(i)$ and a permutation key $k^{(i)}_{prp}$ based on K_{PRP}.
2) Compute the set of r randomly-chosen indices:
$\{I_q \in [1, ..., l] | 1 \le q \le r\}$, where $I_q = k^{(i)}_{prp}(q)$.
3) Calculate the token as:
$v^{(j)}_i = \Sigma^{r}_{q=1} \alpha^{q}_{i} * G^{(j)}[I_q]$, where $G^{(j)}[I_q] = g^{(j)}_{Iq}$.

Note that $v^{(j)}_i$, which is an element of GF(2^p) with small size, is the response the user expects to receive from server j when he challenges it on the specified data blocks.

After token generation, the user has the choice of either keeping the pre-computed tokens locally or storing them in encrypted form on the cloud servers. In our case here, the user stores them locally to obviate the need for encryption and lower the bandwidth overhead during dynamic data operation which will be discussed shortly.The details of token generation are shown in Algorithm 1.

Once all tokens are computed, the final step before file distribution is to blind each parity block $g^{(j)}_i$ in ($G^{(m+1)}$, . . . ,$G^{(n)}$) by
$g^{(j)}_i \leftarrow g^{(j)}_i + f_{kj}(s_{ij})$, i $\in$ {1, . . . , l},
where k_j is the secret key for parity vector $G^{(j)}$ (j $\in$ {m + 1, . . . , n}). This is for protection of the secret matrix P. After blinding the parity information, the user disperses all the n encoded vectors $G^{(j)}$ (j $\in$ {1, . . . , n}) across the cloud servers $S_1, S_2, . . . , S_n$.

3.2 File Retrieval 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 (e.g., r, l, t) appropriately and conducting enough times of verification, we can guarantee the successful file retrieval with high probability.

Algorithm 2 Error Recovery

1: **procedure**
% Assume the block corruptions have been detected among the specified r rows;
% Assume s ≤ k servers have been identified misbehaving
2: Download r rows of blocks from servers;
3: Treat s servers as erasures and recover the blocks.
4: Resend the recovered blocks to corresponding servers.
5: **end procedure**

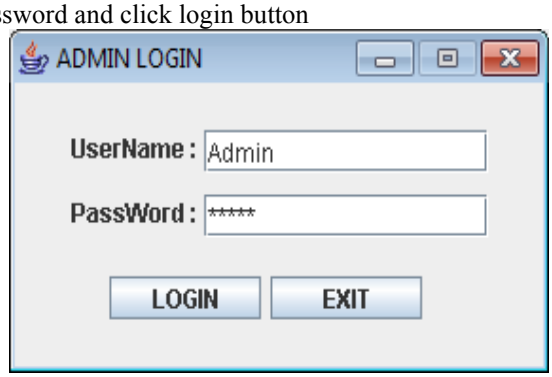
On the other hand, whenever the data corruption is detected, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server(s) (again with high probability). Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and

regenerate the correct blocks by erasure correction, shown in Algorithm 2, 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.

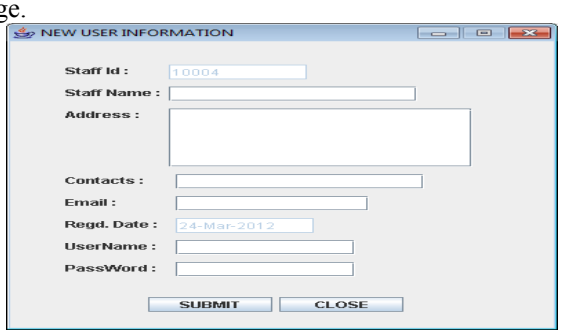
4. IMPLEMENTATION AND RESULTS

Step 1: Start the server

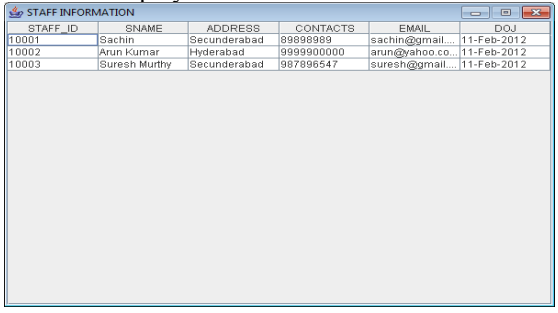
Step 2: Now go to admin page. Enter user name and password and click login button



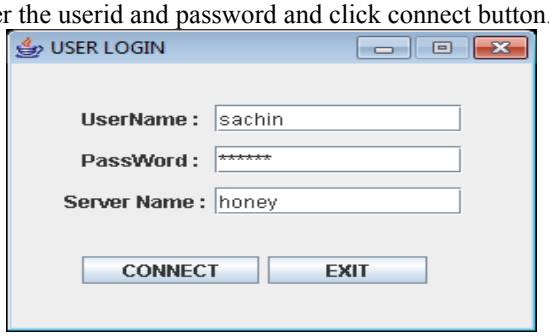
Step 3: Main form will be displayed where the users are created. Create a user by filling the details in the below page.



Step 4: After filling the details clisk submit button. Inorder to view the list of users go to view users option and list of users will be displayed as shown below:

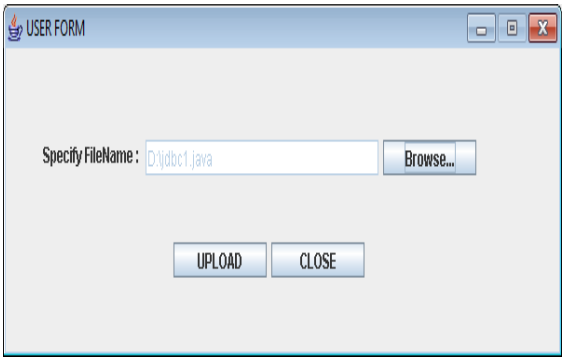


Step 5: Now Start user to perform the transactions. First enter the userid and password and click connect button.



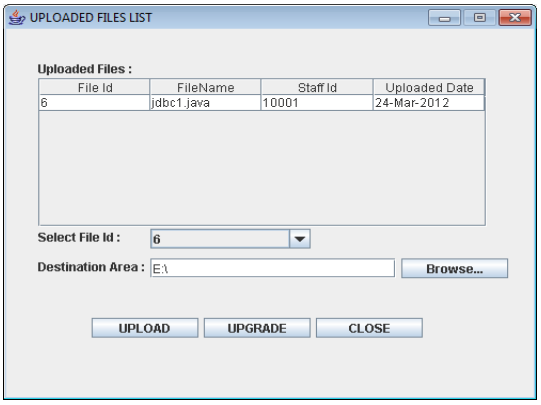
A user form will be displayed. Go to upload option and upload the file

Step 6: Now the user uploads the file by browsing the file from the drive



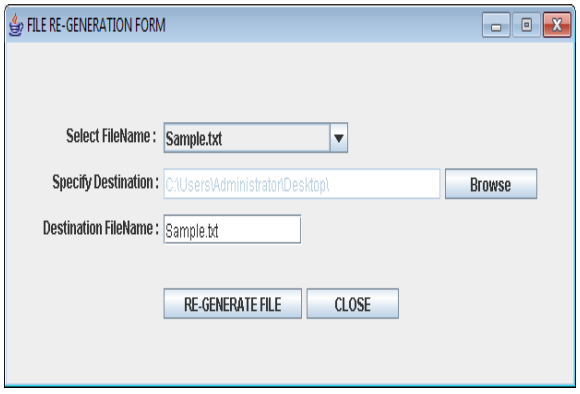
Click on upload button. Now the file will be stored in temp folder

Step 7: Now goto admin page and view the uploaded files. And specify the destination path where the files must be saved



press the upload button, now the file has been moved from temp folder to E drive.

Step 8: If the file has been deleted or modified, there is a chance to regenerate the original file using regenerate option. Enter the filename in regenerate form and specify the destination in order to save the file at that specified location.



Step 9: If the file has been regenerated successfully, a successful message will be displayed as shown below

5. CONCLUSION

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. By utilizing the homomorphic 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 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. C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. of IWQoS'09*, July 2009.
2. C. Wang, K. Ren, W. Lou, and J. Li, "Towards publicly auditable secure cloud data storage services," *IEEE Network Magazine*, vol. 24, no. 4, pp. 19–24, 2010.
3. http://eprint.iacr.org/
4. http://aws.amazon.com/
5. https://www.sun. com/offers/details/sun transparency.xml/
6. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS'09, volume 5789 of LNCS*. Springer-Verlag, Sep. 2009.

Prof Jammi Ashok

Currently working as Professor and Head at Gurunanak Institute of Technology, Hyderabad, A.P, INDIA. He has received his B.E. Degree from Electronics and Communication Engineering from Osmania University and M.E. with specialization in Computer Technology from SRTMU, Nanded, INDIA.

His main research interest includes neural networks, Bioinformatics and Artificial Intelligence. He has been involved in the organization of a number of conferences and workshops. He has been published more than 35 papers in International journals and conferences. He is currently doing his Ph.D from Anna University and submitted thesis.