To Improvise and Design a Cooperative Secure Data Possession Scheme for Integrity Verification for Data Stored on Cloud

Nisha Singh, Dr. Suneetha K.R.

M.tech, Dept of CSE, BIT Bangalore

***Abstract*-- Cloud is emergent field both in terms of high level computing as well as storage. Cloud provides higher computing facilities by clubbing large number of resources online. The cloud storage is concept of storing data online such that data can be accessed by the user at any location and at any hour. Cloud storage has established its importance as organizational as well as individual facility. The storages available with an organization or an individual are usually in form of physical drives which need to be carried to place where they are meant to be used or shared. Cloud storage provides with an opportunity to access the data from anywhere by storing data remotely and making it available for online access. Due to additional overhead involved organizations choose to outsource their data to a cloud service provider. This gives rise to issue of data security in clouds; this includes issues relating to confidentiality, integrity and availability of the data. For tackling this various schemes have been proposed under provable data possession which challenges the server for possession of data that it claims to have. In this paper we propose a cooperative secure data possession scheme which provides an additional layer of security while providing all features of previous data possession scheme.**

Index Terms -- Multi-cloud storage, Web Server based Multi-cloud, Advanced Cooperative Provable Data Possession, Security in clouds.

I. INTRODUCTION

This paper is conceptualized on the need of security of owner's data which is outsourced at the multiple cloud based web servers. For outsourcing the data the most preferable technology is the cloud. Cloud has services on the basis of pay per use. As data owner outsources data to the CSP, the issue of the security arises because misuse of the outsourced data can be done by CSP itself. There are many ways to prevent misuse of data at the small scale like cryptography.

Security is nothing but the intersection of confidentiality, integrity and availability, failing of anyone parameter to achieve leads to data to become vulnerable. Here all the parameters are assessed but the focus is mostly on the issue of integrity. Confidentiality can be achieved through the authentication while the availability can be achieved through the redundant storage of the data on multiple cloud based web servers.

We first consider integrity, which can be achieved by the technique called PDP. In this technique, the data owner challenges to the CSP for providing the guarantee of the integrity of the outsourced data. The challenge is in the form of query. Query contains some credentials of the uploaded data calculated before uploading. Then the CSP calculates the credentials according to the challenge of the data owner in the form of response, the response is given to the owner. Owner crosschecks the original credentials with the response, if equality holds the owner is satisfied with the provided service and integrity the outsourced data. As mentioned above is already existing technique as per paper [7]. But the overhead and the role of the Data Owner is reduced by inducing the third actor in the picture as TTP. All the above stuff taken place in communication in between CSP and Data Owner is happens through TTP. Next we add another layer of security on the owner side to provide extra control over the data that the user stores on the multi-cloud storage.

II. RELATED WORK

2.1 Survey

All material An Ateniese, et al [2] proposed a PDP model which supports problems of static files. This model works optimally for static case with constant complexity by the principle of blocks and tags. It gives the private authentication and the linear storage for Data Owner. But, this model has some limitations which are detected by later results as, expensive server computation, no security guarantee for data possession, vulnerable to replay attacks.

A. Juels, et al [3] proposed a POR model which also works on static storage with the constant complexity. This varies with the previous model in the processing on the data. Data is modified by inserting some sentinel blocks in between which are used to verify the integrity of the file by checking the correctness of sentinel blocks. This model has both communication and computation complexities constant. It has some limitations on number of times one can challenge for integrity as sentinels are one time labels. Also, the model depends on the large preprocessing of data.

Shacham and Waters [4] proposed Compact POR model which is an improved version of POR. This model uses homomorphic property to aggregate a proof of challenge with authenticity complexity of O(1) and with computation complexity of O(t). In this model also some limitations with respect to current models as its solution is for static storage and vulnerable to the leakage of data in verification.

All the above techniques are useful for the static storage only. Ateniese, et al [5] proposed Scalable PDP which is the first technique to give the dynamicity in the outsourcing of data. This model works on the principle of the random oracle model. It takes the pre-computed answers as metadata so it limits on the number of times the updates and the challenges can possible. This model supports only append operation on data and does not provide the in between modifications in the already uploaded data, which also puts limitations on the dynamicity of the model.

C. Erway, et al [6] proposed a DPDP model on the basis of PDP model for dynamic storage of files and also which can be updated online. Even after this modification one can verify the integrity of the file. It uses the skip-lists for maintaining tags and is stored at Data Owner side to avoid replay attacks. This model has computational and communication complexity both up to log(n). The server requires the whole path of data block to access it, because this model does not maintain the numbering to the data blocks. In this case, if the file is too large then both the above complexities are considerable.

Feifei Liu, et al [7] proposed Improved DPDP model which is the improvement over the previous model [6]. It does partition of original file into blocks. Tag is generated for each block and hash value is computed for each tag. These tags are used to verify the integrity of the file by using the skip-lists, and the integrity of the tags is verified by the hash values. In this model computational and communication complexities are reduced from log(n) to constant.

2.2 Motivation

To provide a low cost, scalable, location independent platform for managing client's data. Cloud service providers adopt several distributed file systems for cloud storage systems. These systems provide for storage of data over the cloud. These file systems share some similar features: a single metadata server provides centralized management by a global namespace; files are split into blocks or chunks and stored on block servers; and the systems are comprised of interconnected clusters of block servers. Those features enable cloud service providers to store and process large amounts of data. Hence, it is crucial to offer an efficient verification on the integrity and availability of stored data for detecting faults and automatic recovery. However, they do not provide any mechanisms to check for integrity verification of data as well as the availability of the data.

III. PROPOSED METHODOLOGY

3.1 Architecture

Although existing PDP schemes offer a publicly accessible remote interface for checking and managing the tremendous amount of data, the majority of existing PDP schemes are incapable to satisfy the inherent requirements from multiple clouds in following terms:

The existing protocols suffer from the following weaknesses.

- Most of the data possession schemes do not take dynamic nature of cloud into account.
- Most of the existing techniques provide integrity verification only in post-attack scenario.
- Most of the existing techniques have large upload and download times.
- Most of the existing techniques add overhead on user for pre-processing of file to check for integrity.

To address these problems, we consider a multi-cloud storage service as illustrated in Figure 1. In this architecture, a data storage service involves three different entities: Clients who have a large amount of data to be stored in multiple clouds and have the permissions to access and manipulate stored data; Cloud Service Providers (CSPs) who work together to provide data storage services and have enough storages and computation resources; and Trusted Third Party (TTP) who is trusted to store verification parameters and offer public query services for these parameters.

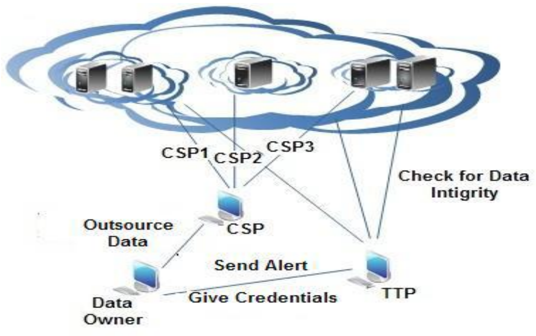

Fig. 1.Verification architecture for data integrity.

In this architecture, we consider the existence ofmultiple CSPs to cooperatively store and maintain theclients' data. Moreover, a cooperative PDP is used toverify the integrity and availability of their stored data in all CSPs. The verification procedure is described as follows: Firstly, a client (data owner) uses the secret key to pre-process a file which consists of a collection of .. blocks, generates a set of public verification information that is stored in TTP, transmits the file and some verification tags to CSPs, and may delete its local copy; Then, by using a verification protocol, the clients can issue a challenge for one CSP to check the integrity and availability of outsourced data with respect to public information stored in TTP. We neither assume that CSP is trust to guarantee the security of the stored data, nor assume that data owner has the ability to collect the evidence of the CSP's fault after errors have been found. To achieve this goal, a TTP server is constructed as a core trust base on the cloud for the sake of security. We assume the TTP is reliable and independent through the following functions [8]: to setup and maintain the CPDP cryptosystem; to generate and store data owner's public key; and to store the public parameters used to execute the verification protocol in the CPDP scheme. Note that the TTP is not directly involved in the CPDP scheme in order to reduce the complexity of cryptosystem.

3.2 CSDP Algorithm

Step 1: Input file F

Step 2: Using Hash Index Hierarchy concept to split the file.

Step 3: Split file into number of blocks {m1, m2, m3,….}

Step 4: Generate key [7] for each block
Use user's key(sk) applied for keygen(sk,pk)
Where,
sk= encryption of key choosen by user
pk= generated from sk , $0<pk<10^6$

Step 5: Generate Tag [7] for each block
TagBlock(sk, pk, mi, vi, i) →{Ti, hi}
Compute, $T^*_i = g^{mi} \bmod N$
$H^*_i = H_{k1}(Ti \,\|\, f(vi) \,\|\, i)$
Where,
H=cryptographic hash function
f = pseudo random function

Step 6: Store all the credentials in the form of tags and hash on the TTP before uploading the data on clouds.

Step 7: Response calculation by TTP from clouds:
Challenge: Query(c) →chal [7]
i. Input: number of blocks to be challenged,b
ii. k2, k3 → random numbers.
index: $i_j = \pi_{k2}(j)$ for $1 \le j \le b$
Coeff.:$a_j = \pi_{k3}(j)$; π = pseudo random permutation
iii. Output: chal = $\{(i_1, i_2, \ldots, i_b), (a_1, a_2, \ldots, a_b)\}$

Proof: Prove(chal, F) → P [7]
i. Input: Query chal, File F
ii. Search h_{ij}, T_{ij}, m_{ij}For $i_j \in \{ i_1, i_2, \ldots, i_b\}$

Use the HVR concept [1] to integrate the responses from multiple clouds.
Homomorphism mapped as, f: P → Q
Where, P and Q are two different groups
Such that,
$f(g1 \oplus g2) = f(g1) \otimes f(g2)$; for all g1, g2 $\in$ P
Where,
$\oplus$ = operation in P
$\otimes$ = operation in Q

The above homomorphism concept is used for calculating Homomorphic Verifiable Tags such that, T_i and T_j for messages m_i and m_j respectively, then by combining T_i and T_j get T' for the $m_i + m_j$.

By working on the same principle the HVR can be calculated from different responses from different clouds in CSDP scheme and stored at TTP.
Compute, $M=a_1.m_{i1} + a_2.m_{i2} + \ldots + a_c.m_{ib}$
$h= hi_1. hi_2 \ldots hi_b$
iii. Output: P= $\{M, h, (Ti_1, Ti_2 \ldots Ti_b)\}$

Step 7: Verify the collective response with the RESPONSE stored at TTP.

Verification:
Verify(s_k, chal, P,Ω) → {Accept, Reject}

i. Compute, $T^* = g^m \bmod N$
$T=T_{i1}^{a1} * T_{i2}^{a2} * \ldots * T_{ib}^{ab}$
Search, $\Omega=\{v_{i1}, v_{i2}, \ldots, v_{ib}\}$ for $i_1 \ldots i_b$
$H=H_{k1}(T_{i1}\|f(v_{i1})\|i_1) * \ldots * H_{k1}(T_{ib}\|f(v_{ib})\|i_b)$
ii. Check,
T* ?= T, & H*? = h
Where, T* and H* are nothing but the original credentials.
If both holds, output Accept;
Else, output Reject.
Step 8: If there is mismatch, generate alerts.
Step 9: Otherwise, SUCCESS.

IV. CONCLUSION

In this paper, we presented the construction of an efficient cooperative secure data possession scheme for distributed cloud storage. Based on homomorphic verifiable response and hash index hierarchy, we have improvised a scheme to support dynamic scalability on multiple storage servers. We also showed that our scheme provided all security properties required by zeroknowledge interactive proof system, so that it can resist various attacks even if it is deployed as a public audit service in clouds. In this paper we were also able to give another layer of security including the TTP's so the hacker or attacker needs to accomplish a triple layer breach before reaching the client's data.

In future work the scheme could be improved to use higher mechanisms for security enhancements. Also the scheme is dependent on Hash index hierarchy and it needs to be able accommodate requirements of two-layer architecture.

V. REFERENCES

1] Yan Zhu, Hongxin Hu, Gail-JoonAhn, Mengyang Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23, NO. 12, DECEMBER 2012.

[2] G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J. Peterson, and D.X. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 598-609, 2007.

[3] A. Juels and B.S.K. Jr., "Pors: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, 2007.

[4] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08), pp. 90-107, 2008.

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

[6] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession", In CCS '09, pp. 213-222, April 24,2012.

[7] Feifei Liu, DavuGu, HainingLu,"An Improved Dynamic Provable Data Possession", Proceedings of IEEE CCIS2011, pp 290-295, 2011.

[8] B. Sotomayor, R.S. Montero, I.M. Llorente, and I.T. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," IEEE Internet Computing, vol. 13, no. 5, pp. 14-22, Sept. 2009.