Identity-Based Proxy-Oriented Data Uploading and Remote Data Integrity Checking in Public Cloud

B.Abinaya

Department of CSE
IFET College of Engineering
Villupuram,India
abinaya191995@gmail.com

Mrs.S.G.Sandhya

Department of CSE
IFET College of Engineering
Villupuram,India
sgsandhyadhas@gmail.com

ABSTRACT----**Present days many users store their significant data in cloud. To ensure that the security of the cloud stored data users need to encrypt the important data. The point of data security which has always been noteworthy aspect of quality, cloud computing cause a new security threats. In cloud storage systems, the server that stores the client's data is not necessarily trusted. The existing system does not provide security mechanisms for storing data in clouds, but the propose system can provide security against for Collusion attack, DDOS attack. Existing data auditing schemes have security risks in processing of data. For achieving the efficiency of cloud storage, the proposed system provides flexible data segmentation with additional authorization process among the three participating parties of client, server and a third-party auditor (TPA). We propose an identity based data storage scheme, it will resist the collusion attacks.**

Keywords-collusion;authorization;segmentation;auditor

I. INTRODUCTION

The Identity based scheme provides efficient dynamic data operations for data in cloud computing. This is because user wishes to do various block level operation on the data file by assuring the data integrity. It assumes that CSS will provide the correct data to user without deceiving the user. The block Level operation performed in fine grained updates. To achieve this, this scheme utilizes a flexible data segmentation strategy and a data auditing protocol. The data segmentation is the way of splitting the whole file into countable number of parts and are stored in different server locations. This technique is done for data security. The adversary does not know the file locations of various fragmented parts of file. Thus, he cannot view the whole collected single file. Thus we can protect the data. Meanwhile, it address a potential security problem in supporting public verifiability to make the scheme more protected and robust, which is achieved by adding an additional authorization process among the three partaking parties of client, server and a Manager. For better security, our scheme incorporates an additional authorization process with the aim of eradicating threats of unauthorized audit challenges from malicious or pretended third-party auditors, which we term as 'authorized auditing'. Thus, the segmented files are encrypted and stored in different server locations for enhancing the security purposes. Also only authorised persons are allowed to access the data. There are three main functions that are performed here are:

A. Uploading the file:

The data owner will create a mail ID and tries to store the file in cloud. He will use the mail ID and logins the cloud. Then uploads the file by getting acceptance from the server admin. After the admin accepts the request, the data owner will try to upload the file. Firstly, the owner will select the file. Then, he will segment the file into various parts. Then, these segmented parts are encrypted and stored in various server locations.

B. Server admin acceptance:

Once the data owner requests the server admin for uploading the data, then he will either accept the request or reject it based on the authorisation.

C. Data retrival:

The data user, if wishes to access a file, then he will logon the cloud with his mail ID and types a keyword in file. The system will sort out the file. He wil get the acceptance from the acceptance to decrypt the data, he will retrieve the data.

II. RELATED WORKS

The verification for proper data tends to be very simple, so that unauthorized person sends auditing service message to server. This creates many issue like distributed denial of service. The adversary can get sensitive information. They also support full data dynamics but unsuitable for variable sized blocks. In the rank based Merkle Hash Tree each node N will have a maximum of 2 child nodes. In fact, according to the update algorithm, every non-leaf node will constantly have 2 child nodes. Each child nodes have varied in their block size. The time for retrieving the data from the block may vary according to the size of the block. If any user wants to update their file in the block, the server will return the block which is unstayed for the long time. In outsourcing services and resource sharing network services, the user can store datas and share datas. But, as they are third party services, these services are lagging in security. This lead to the evolution of the provable data possession (PDP) model. By this model, the data owner will preprocesses the data before outsourcing them. Later this is stored in the cloud. The client later proves the server that he is an authorized user and then he also asks to the server that to prove that the data has integrity property without downloading them. Later the model is constructed with an efficient framework called dynamic provable data possession (DPDP), which supports the user to add up or updates the data in already stored data dynamically, by simply adding or updating the contents in the stored file instead of storing the entire file again. This is an efficient way to store the data in cloud with ease of work, minimizes the cost consumption and time consumption. The main aim of this model is to reduce the space in server. Also these services are extended to have a copy of resource by means of the multiple replica provable data possession (MRPDP) model. This model is used to ensure the resource maintenance by having a copy of resources in server and when one file is damaged, another file can retrieve the contents of the file. The auditing services are important to ensure whether the data owner is storing the useful resources in the cloud server or not. This is maintained by mean of the third-party auditor (TPA). This third-party auditor will ensure this services. Also multiple auditing tasks can be done at the same time in public cloud services, as they have many users. They extend the efficiency and security of public cloud services. Also they work dynamically in order to maintain the system dynamically, as there are many updates are carried out at a time in public cloud services.

III. PROPOSED SYSTEM

This Identity based scheme provides efficient dynamic data operations for data in cloud computing. This is because user wishes to do various block level operation on the data file by assuring the data integrity. It assumes that CSS will provide the correct data to user without duplicitous the user. The block Level operation performed in fine grained Updates.To achieve this, this scheme utilizes a flexible data segmentation approach and a data auditing protocol. In the meantime, it address a potential security problem in supporting public verifiability to make the scheme more safe and forceful, which is achieved by adding an additional authorization process among the three participating parties of client, server and a Manager. For better security, our scheme combines an additional authorization process with the aim of rejecting threats of unauthorized audit challenges from cruel or pretended third-party auditors, which we term as ‘authorized auditing’.

The various modules used here are:

a) Setup and data upload:

In cloud, user data is kept remotely on CSS. In order to confirm the data without regaining them, the client will need to prepare verification metadata. Then, these metadata will be uploaded and stored alongside with the unique datasets. These tags are designed from the original data; they must be small in size in comparison to the original dataset for practical use.

The system architecture is given by:

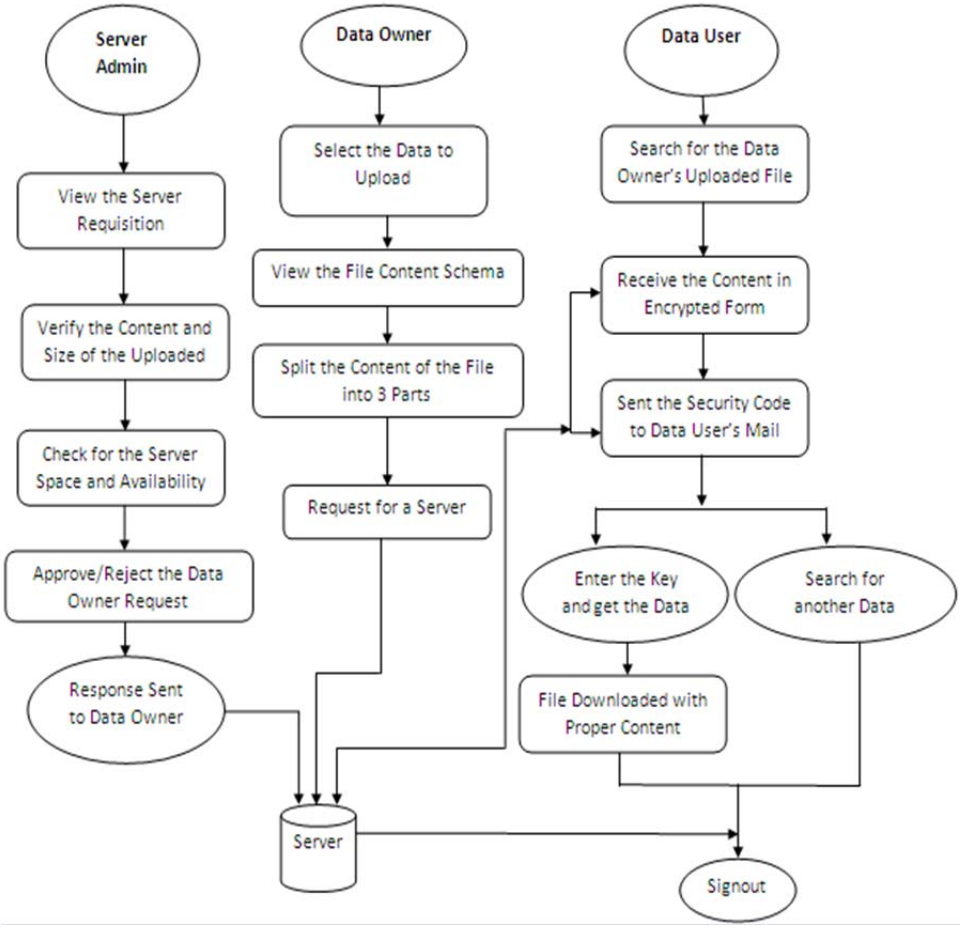

Fig.3.1.System Architecture Design

b) Authorization for TPA:

This Module is not required in a two-party scenario where clients verify their data for themselves, but it is important when users require a semi-trusted TPA to verify the data on their behalf. If a third party can enormously ask for integrity evidences over a certain piece of data, there will always be security risks in existence such as plaintext extraction.

c) Verification of data storage:

This Module is where the main requirement integrity verification to be fulfilled. The client will send a challenge message to the server, and server will compute a response over the pre-stored data and the challenge message. The client can then verify the response to find out whether the data is intact. The scheme has public verifiability if this verification can be completed without the client's secret key. If the data storage is static, the total process would have been ended here.

d) Data update:

Befalls in dynamic data backgrounds. The client needs to perform updates to some of the cloud data storage. The updates could be roughly categorized in insert, delete and modification; if the data is deposited in blocks with varied size for efficiency reasons, there will be more types of apprises to address.

e) Metadata update:

In order to keep the data storage stay verifiable lacking retrieving all the data stored and/or re-running the whole setup phase, the client will essential to update the verification metadata, conferring with the existing keys.

f) Verification of updated data:

This is also an vital step in dynamic data context. As the CSS is not totally confidential, the client needs to verify the data update process to see if the updating of both user data and verification metadata have been done fruitfully in order to ensure the updated data can still be verified correctly in the future.

Techniques used:

The system model and security explanation are existing in this section. An ID-DPDP protocol includes four different entities. Described as below:

a) Client: an entity, which has massive data to be deposited on the multi-cloud for preservation and calculation, can be either individual consumer or corporation.

b) CS (Cloud Server): an entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data.

c) Combiner: an entity, which receives the storage request and distributes the block-tag pairs to the corresponding cloud servers. When getting the challenge, it separates the challenge and distributes them to the different cloud servers. When receiving the responses from the cloud servers, it combines them and sends the combined response to the verifier.

d) PKG (Private Key Generator): an entity, when receiving the identity, it outputs the private key.

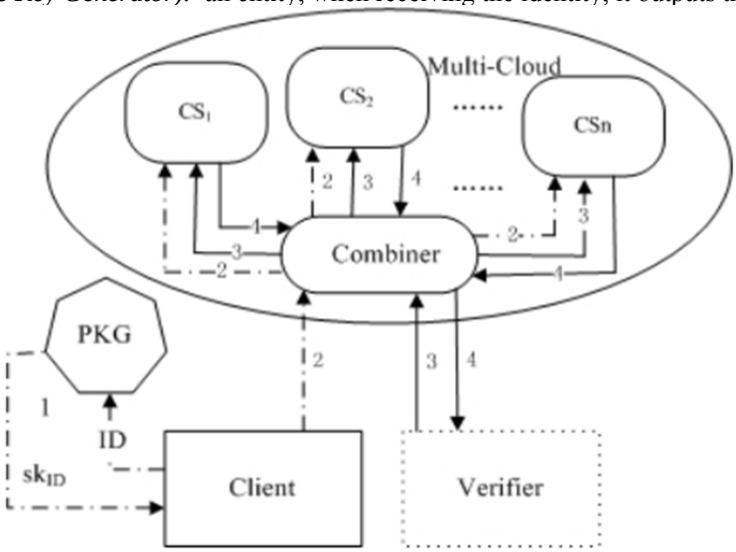

Fig.3.2.Architecture of ID-DPDP Protocol

This protocol includes four procedures: Setup, Extract, TagGen, and Proof. Its architecture can be depicted in Figure. The figure can be described as follows: 1. In the phase Extract, PKG generates the private key for the client. 2. The client produces the block-tag pair and uploads it to syndicate. The combiner issues the block-tag pairs to the different cloud servers according to the storage metadata. 3. The verifier directs the challenge to combiner and the combiner issues the contest query to the corresponding cloud servers according to the storage metadata. 4. The cloud servers reply the challenge and the combiner sums these responses from the cloud servers. The combiner sends the aggregated response to the verifier. Finally, the verifier checks whether the aggregated reply is effective.

The concrete ID-DPDP construction mainly comes from the signature, provable data possession and distributed computing. The signature narrates the client's identity with his private key. Distributed computing is used to store the client's data on multi-cloud servers. At the same time, distributed computing is also used to syndicate the multi-cloud servers' replies to return the verifier's test. Based on the provable data possession protocol, the ID-DPDP protocol is constructed by making use of the signature and distributed computing.

Without loss of generality, let the number of stored blocks be n. For different block Fi , the corresponding tuple (Ni , CSli , i) is also different. Fi denotes the i-th block. Denote Ni as the name of Fi . Fi is stored in CSli where li is the index of the corresponding CS. (Ni , CSli , i) will be used to generate the tag for the block Fi . The algorithms can be described in detail below:

- *Setup:*

Let g be a generator of the group G1 with the order q. Define the following cryptographic hash functions:

H : {0, 1} ∗ → Z∗ q h : {0, 1} ∗ × Z∗ q → G1 h1 : {0, 1} ∗ → Z∗ q

Let f be the pseudo-random function and π be a pseudorandom permutation. They can be described in detail below:

f : Z ∗ q × {1, 2, · · · , n} → Z∗ q

π : Z ∗ q × {1, 2, 3,· · · , n} → {1, 2, 3, · · · , n}

PKG choices a random number x ∈ Z∗ q and calculates Y = g x . The parameters {G1, G2, e, q, g, Y, H, h, h1, f, π} are made public. PKG keeps the master secret key x confidential.

- *Extract:*

Input the identity ID, PKG picks r ∈ Z∗ q and calculates

R = g r , σ = r + xH(ID, R) mod q

PKG sends the private key skID = (R, σ) to the client by the protected channel. The client can confirm the correctness of the received private key by checking whether the following equation holds.

g σ = R YH(ID,R) …… (1)

If the formula (1) clamps, the client ID receives the private key; else, the client ID rejects it.

- *TagGen(skID, F,P):*

Split the whole file F into n blocks, i.e.,

F = (F1, F2, · · · , Fn). The client makes to stockpile the block Fi in the cloud server CSli . Then, for 1 ≤ i ≤ n, each block Fi is split into s sectors, i.e.,

Fi = {F˜ i1, F˜ i2, · · · , F˜ is}. Thus, the client gets n × s sectors {F˜ ij}i∈[1,n],j∈[1,s] . Picks s random u1, u2, · · · , us ∈ G1.

Denote u = {u1, u2, · · · , us}. For Fi , the client performs the procedures below:

1) The client calculates

Fij = h1(F˜ ij) for every sector F˜ ij , 1 ≤ j ≤ s.

2) The client calculates

Ti = (h(Ni , CSli , i) Ys j=1 u Fij j) σ

3) The client adds the record

φi = (i, u, Ni , CSli) to the table Tcl. It stores Tcl locally.

4) The client sends the metadata table Tcl to the combiner. The combiner adds the records of Tcl to its own metadata table To.

5) The client outputs Ti and stores (Fi , Ti) in CSli .

- *Proof (P, C, V):*

This is a 5-move protocol among P = {CSi}i∈[1,nˆ]), C, and V with the input (public parameters, Tcl). If the client delegates the verification task to some verifier, it sends the metadata table Tcl to the verifier. Of course, the verifier may be the third auditor or the client's proxy. The interaction protocol can be given in detail below.

1) Challenge 1 (C ← V): the verifier picks the challenge chal = (c, k1, k2) where 1 ≤ c ≤ n, k1, k2 ∈ Z∗ q and sends chal to the combiner C;

2) Challenge 2 (P ← C): the combiner calculates vi = πk1 (i), 1 ≤ i ≤ c and looks up the table To to get the records that correspond to {v1, v2, · · · , vc} = M1 SM1 S · · · SMnˆ. Mi denotes the index set where the corresponding block-tag pair is stored in CSi . Then, C sends (Mi , k2) to CSi ∈ P.

3) Response1 (P → C): For CSi ∈ P, it performs the procedures below:

A. For vl ∈ Mi , CSi splits Fvl into s sectors Fvl = {F˜ vl1, F˜ vl2, · · · , F˜ vls} and calculates Fvl j = h1(F˜ vl j) for 1 ≤ j ≤ s. (Note: Fvl j = h1(F˜ vl j) can also be precomputed and stored by CS)

B. CSi calculates al = fk2 (l), vl ∈ Mi and T (i) = Y vl∈Mi Tvl al

C. For 1 ≤ j ≤ s, CSi calculates Fˆ (i) j = X vl∈Mi alFvl j Denote Fˆ(i) = (Fˆ (i) 1 , · · · , Fˆ (i) s). d) CSi sends θi = (Fˆ(i) , T (i)) to C.

4) Response2 (C → V): After receiving all the responses from CSi ∈ P, the combiner aggregates {θi}CSi∈P into the final response as T = Y CSi T (i) , Fˆ l = X CSi Fˆ (i) l Denote Fˆ = (Fˆ 1, Fˆ 2, · · · , Fˆ s). The combiner sends θ = (F , T ˆ) to the verifier V .

5) After receiving the response

θ = (F , T ˆ), the verifier calculates

vi = πk1 (i)
hi = h(Nvi , CSlvi , vi)
ai = fk2 (i)

Then, it verifies whether the following formula holds.

e(T, g) =e(Yc i=1 h ai i Ys j=1 u Fˆj j , RY H(ID,R)) …… (2)

If the formula (2) holds, then the verifier outputs "success". Otherwise, the verifier outputs "failure".

In Proof(P, C, V), the verifier picks the challenge chal = (c, k1, k2) where k1, k2 are randomly chosen from Z∗ q . Based on the challenge chal, the combiner determines the challenged block index set as {v1, v2, · · · , vc} where vi = πk1 (i), 1 ≤ i ≤ c. Since πk1 (·) is a pseudo-random permutation determined by the random k1, the challenged c block-tag pairs come from the random selection of the n stored block-tag pairs.

An ID-DPDP protocol must be workable and correct. That is, if the PKG, C, V and P are honest and follow the specified procedures, the response θ can pass V 's checking. The correctness follows from below:

e(T, g)
= e(Q CSi T (i) , g)
= e(Q CSi Q vl∈Mi T fk2 (l) vl , g)
= e(Q CSi Q vl∈Mi (h al l Qs j=1 u alFvl j j), gσ)
= e((Qc i=1 h ai i) Qs j=1 u Fˆj j , RY H(ID,R))

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song; "Provable Data Possession at Untrusted Stores," CCS'07, pp. 598-609, 2007.

[2] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik; "Scalable and Efficient Provable Data Possession," SecureComm 2008, article 9, 2008.

[3] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia; "Dynamic Provable Data Possession" , CCS'09, 213-222, 2009.

[4] F. Seb´e, J. Domingo-Ferrer, A. Mart´ınez-Ballest´e, Y. Deswarte, J. Quisquater; "Efficient Remote Data Integrity checking in Critical Information Infrastructures. IEEE Transactions on Knowledge and Data Engineering," 20(8):1034-1038, 2008.

[5] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, S. S. Yau; "Efficient Provable Data Possession for Hybrid Clouds," CCS'10, 756-758, 2010.

[6] Y. Zhu, H. Hu, G.J. Ahn, M. Yu; "Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage," IEEE Transactions on Parallel and Distributed Systems, 23(12):2231-224, 2012.

[7] R. Curtmola, O. Khan, R. Burns, G. Ateniese. MR-PDP; "Multiple-Replica Provable Data Possession," ICDCS'08, 411-420,2008.

[8] C. Wang; "Toward publicly auditable secure cloud data storage services," IEEE Network, vol. 24, no. 4, pp. 19 24, 2010.

[9] Q. Wang, C.Wang, K. Ren,W. Lou, and J. Li; "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5,pp. 847-859, May 2011.

[10] X. Zhang, C. Liu, S. Nepal, S. Panley, and J. Chen; ''A Privacy Leakage Upper-Bound Constraint Based Approach for Cost- Effective Privacy Preserving of Intermediate Datasets in Cloud," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 6, pp. 1192-1202, June 2013.