Implementing Proof of Retriavaibility for Multiple Replica of Data File Using No-SQL Database

M. E. Pore, S. B. Takmare, Amol Mahadik

[1]Computer Science and Engineering Dept., Bharati Vidyapeeth's College of Engineering, Kolhapur (District), Maharashtra-416012, INDIA maheshpore17@gmail.com
[2]Head of Department, Computer Science and Engineering Dept., Bharati Vidyapeeth's College of Engineering, Kolhapur (District), Maharashtra-416012, INDIA sachintakmare@gmail.com
[3] Lecturer, Computer Engineering Dept., Bharati Vidyapeeth's Institute of Technology, Palus, Sangli (District), Maharashtra-416310, INDIA mahadik.amol@gmail.com

---***---

Abstract - *Cloud computing comprises high demand as an outsourcing entity to share storage capacity, different vendors databases, a pool of software's and different platforms. Instead of wasting money, many organizations outsource their services from different Cloud service providers. One of the key services of Cloud service provider is providing storage capacity, which is in high demand now a day. But, Cloud computing is in threat when number of users increased. The Cloud service provider may behave unfaithfully towards user in case of CSP with limited resources and many users. This paper designs a protocol to check integrity of data stored at Cloud server. Many schemes introduce a new network entity 'Third Party Auditor'. It reduces the integrity verification burden of a user. The TPA performs integrity verification of data stored at cloud server on behalf of user. However, it is not enough to trust on TPA in case of sensitive data. The TPA may also behave unfaithfully towards user. So, this paper proposes a new protocol that generates a proof of data integrity in case of untrusted cloud storage as well as untrusted TPA. Furthermore, to work with and process user's time series data this paper replaces relational database system with No-SQL databases. This will increase the data operation performance in case of huge data management.*

Keywords: Cloud computing, No-SQL database, Untrusted Third Party Auditor, Integrity of data.

1. INTRODUCTION

Cloud computing is a model to compute mobile, suitable and as per need access to resources shared at centralized place. Cloud computing effectively utilizes the IT resources. Cloud computing provides different services like Iaas, Paas and Saas. One of the major role in Iaas is to provide storage as a service. This reduces the burden of the user to store data locally. It is safe to utilize this service only if when numbers of users are limited. In case of Cloud service provider with resource constrained devices, if the number of users increased the CSP may behave unfaithfully towards user. Some user's store their sensitive data at CSP's, so checking the correctness of user data and providing safeguards from unauthorized access is most essential one. The data may be corrupted or losses due to certain server breakdowns or hacked by intruders or may be intentionally deleted by CSP [1].

In case of CSP with resource constrained devices, to increase resource utilization CSP itself intentionally deletes user's rarely accessed data. So now a day in such cases maintaining verification of user data through various protocols is in focus. The many schemes implements integrity proofs through different integrity proofs techniques [2], [3], [4], [5], [6], [7], [8], [9], [10] and [11].

To generates integrity proofs large number of computations are required. It creates a huge overhead of creation of authenticators, tagging of data, and creation of secret keys at user side. The public verifiability reduces this computational overhead of user by appointing Third Party Auditor. The TPA executes different data structure and algorithms to generate data integrity proofs.

As a safeguard to user data, the data is encrypted by using set of secret keys at different time periods. The TPA is always has the possession with this secret key. It may be possible that TPA also modifies or deletes some data accidentally or intentionally. In this case generated integrity proof is not correct, so ensuring TPA integrity proofs is essential. This paper proposes a protocol that verifies TPA's generated integrity proofs.

Now a day due to real time data large data warehouses are require to store it. The CSP's may maintain this data in distributed manner. This also requires greater data operation performance as well as scaling of data at run time. All this is not possible with traditional relational database management system.

This paper replaces traditional relational database management system by No-SQL database. It provides Decentralized access to data, maintains replicas of dataset which produces availability, data scalability and stores large volume of data. All this are very helpful for cloud applications [14], [15], [16], [17], [18] and [19].

By taking all these in consideration this paper proposes a protocol that checks integrity of user data in case of untrusted CSP as well as untrusted TPA and in No-SQL environment.

2. LITERATURE REVIEW

Jin Li, Xiao Tan, Xiaofeng Chen, Duncan S. Wong, and Fatos Xhafa propose the notion of public verifiability to reduce the computational cost at user side during the integrity verification of their data. They introduce one more network element which is acting as an auditor on the behalf of the client. The auditor preprocesses client's data and uploads it to the storage server and later verifies the integrity of data. They were formerly proposed integrity of only a single copy of a data file by using public verifiability. They also assumed that a cloud audit server is trusted network element which will fail at some incidents. They used a relational database system which has some database scalability issues [1].

Ateniese et al. [2] defined the model for ensuring possession of files on untrusted storages. They also proposed the first proof-of-storage scheme that supports public verifiability. The scheme utilizes RSA-based homomorphic tags for auditing outsourced data, However, the data owner has to compute a large number of tags for those data to be outsourced, which usually involves exponentiation and multiplication operations. Furthermore, the case of dynamic data storage has not been considered by Ateniese et al.

Juels and Kaliski Jr. [3] introduced spot-checking and error correcting codes are adopted to ensure both "possession" and "retrievability" of data files in archive service systems. However, public verifiability is not supported in their scheme and the data owner also has to make many computational efforts to generate tags for those data to be outsourced.

Much research devoted their work in data integrity and security in cloud computing for stored data [4], [5], [6], [7], [8], [9], [10] and [11].

Jia Yu, Kui Ren, Cong Wang and Vijay Varadharajan discuss a new aspect of cloud auditing. They investigate how to reduce the damage of the client's key exposure in cloud storage auditing. They formalize the definition and the security model of the auditing protocol with key-exposure resilience and propose such protocol. However, they are not applying this technique in the case of third party auditor used for cloud storage auditing. Again, public verifiability is not supported in their scheme and the data owner also has to make many computational efforts [12].

Zhuo Haoand Nenghai Yu enlightens a multiple-replica remote data possession checking protocol which has public verifiability. Homomorphic authentication tags based on BLS signature are used in the proposed protocol. Again, they used a relational database system to maintain the database of multiple replicas, which required a large number of complex operations with large data sets [13].

Ruxandra Burtica, Eleonora Maria Mocanu, Mugurel developed an application which follows a keyword over multiple social media platforms (e.g. Twitter, Facebook), maintaining the aggregated data in a No-SQL database [14].

Jing Han, Haihong E, Guan Le and Jian Du define basic characteristics, the data model of No-SQL. In addition, they classify No-SQL databases according to the CAP theorem. Finally, the mainstream No-SQL database is separately described in detail, and defines some properties to help enterprises to choose No-SQL [15].

R.Hecht and S.Jablonski, defines use of No-SQL in Cloud Computing [18].

3. DESIGN ASPECTS

3.1 Proposed System

This paper designs and develops a verification scheme with public verifiability so that the client can verify correctness of the remote data using No-SQL database. In addition, exploring the auditing protocol with key-exposure resilience to prevent dishonest behavior of Third Party Auditor (Cloud Audit Server) and reduce computation and communication overhead.

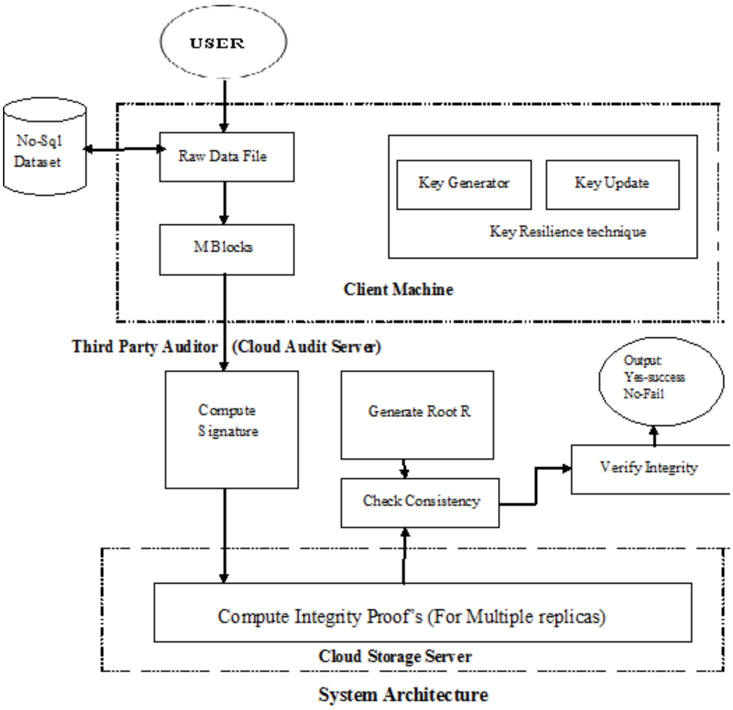


Fig - 1: Architecture of Proposed System

3.2 Methodology

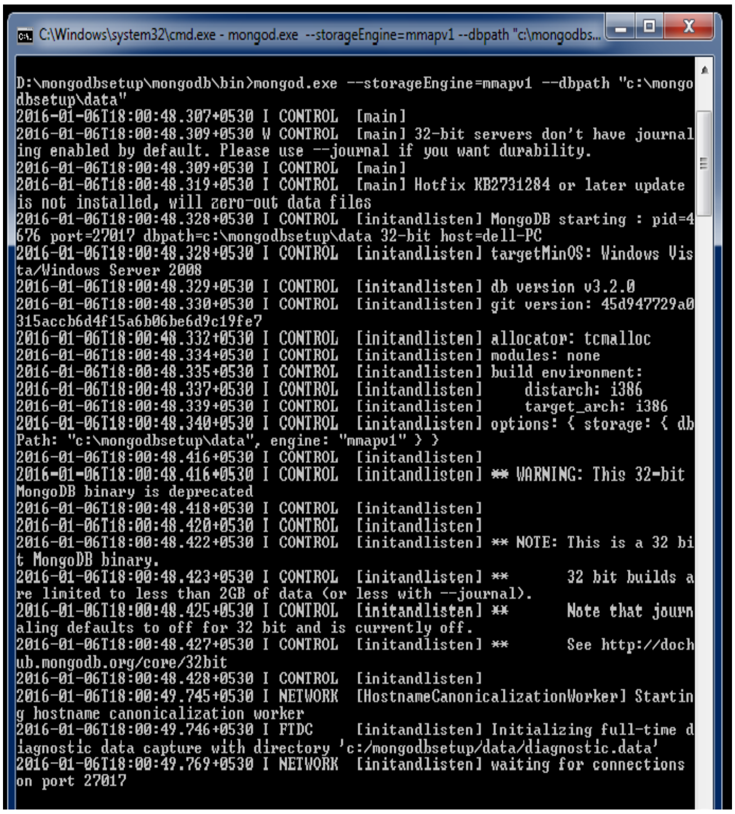
Step 1: **Data collection and Setup module:** In Data Collection the traditional data warehouse solution replaced into emerging big data technologies such as No-SQL databases. So the client's data are stored into No-SQL type databases, which is a document/text/XML like structure. By using Searching technique user can collect a data file from database and divide it into the M blocks and send these blocks to Cloud Audit Server. In addition to this, user generates set of secret keys used at different time period to encrypt the data timely to provide safeguard.

Step 2: Upload module: - In the second module the user uploads Dataset blocks to the Cloud Audit server. The Cloud Audit Server generates a signature for each dataset blocks and re-uploads the data blocks to the Cloud Storage Server.

Step 3: Signature generation Module: - For each data block signature is generated by TPA. All these signature set is then also sent to the client. Integrity proofs are calculated from these signatures.

Step 4: Integrity verification module: - The TPA challenges the CSS for integrity verification of data file. In response to this CSS provides proofs, if both matches then data are correct otherwise data is corrupted/modified or lost by CSS. The same is done by user to check proofs given by TPA. All these challenge and proof are done for multiple replicas of data file.

4. SNAPSHOTS

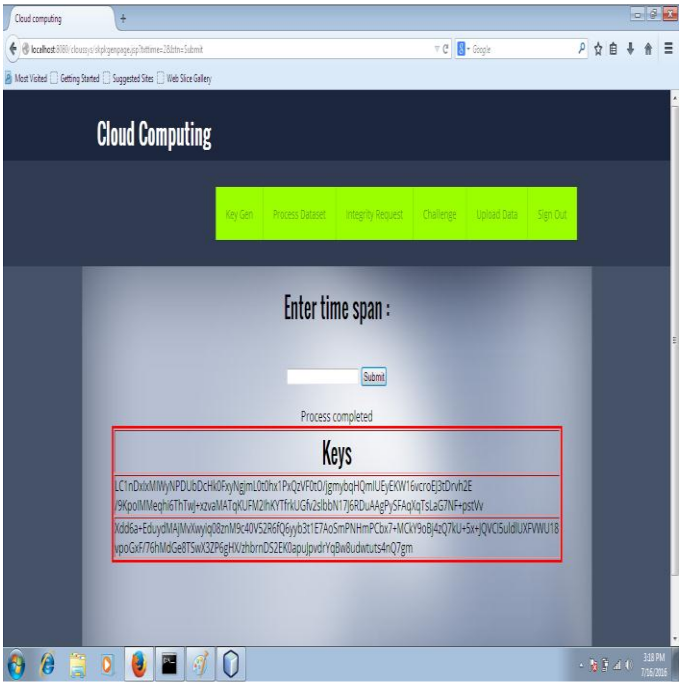


C:\Windows\system32\cmd.exe - mongod.exe --storageEngine=mmapv1 --dbpath "c:\mongodbs...

```
D:\mongodbsetup\mongodb\bin>mongod.exe --storageEngine=mmapv1 --dbpath "c:\mongo
dbsetup\data"
2016-01-06T18:00:48.307+0530 I CONTROL  [main]
2016-01-06T18:00:48.309+0530 W CONTROL  [main] 32-bit servers don't have journal
ing enabled by default. Please use --journal if you want durability.
2016-01-06T18:00:48.309+0530 I CONTROL  [main]
2016-01-06T18:00:48.319+0530 I CONTROL  [main] Hotfix KB2731284 or later update
is not installed, will zero-out data files
2016-01-06T18:00:48.328+0530 I CONTROL  [initandlisten] MongoDB starting : pid=4
676 port=27017 dbpath=c:\mongodbsetup\data 32-bit host=dell-PC
2016-01-06T18:00:48.328+0530 I CONTROL  [initandlisten] targetMinOS: Windows Vis
ta/Windows Server 2008
2016-01-06T18:00:48.329+0530 I CONTROL  [initandlisten] db version v3.2.0
2016-01-06T18:00:48.330+0530 I CONTROL  [initandlisten] git version: 45d947729a0
315accb6d4f15a6b06be6d9c19fe7
2016-01-06T18:00:48.332+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2016-01-06T18:00:48.334+0530 I CONTROL  [initandlisten] modules: none
2016-01-06T18:00:48.335+0530 I CONTROL  [initandlisten] build environment:
2016-01-06T18:00:48.337+0530 I CONTROL  [initandlisten]     distarch: i386
2016-01-06T18:00:48.339+0530 I CONTROL  [initandlisten]     target_arch: i386
2016-01-06T18:00:48.340+0530 I CONTROL  [initandlisten] options: { storage: { db
Path: "c:\mongodbsetup\data", engine: "mmapv1" } }
2016-01-06T18:00:48.416+0530 I CONTROL  [initandlisten]
2016-01-06T18:00:48.416+0530 I CONTROL  [initandlisten] ** WARNING: This 32-bit
MongoDB binary is deprecated
2016-01-06T18:00:48.418+0530 I CONTROL  [initandlisten]
2016-01-06T18:00:48.420+0530 I CONTROL  [initandlisten]
2016-01-06T18:00:48.422+0530 I CONTROL  [initandlisten] ** NOTE: This is a 32 bi
t MongoDB binary.
2016-01-06T18:00:48.423+0530 I CONTROL  [initandlisten] **       32 bit builds a
re limited to less than 2GB of data (or less with --journal).
2016-01-06T18:00:48.425+0530 I CONTROL  [initandlisten] **       Note that journ
aling defaults to off for 32 bit and is currently off.
2016-01-06T18:00:48.427+0530 I CONTROL  [initandlisten] **       See http://doch
ub.mongodb.org/core/32bit
2016-01-06T18:00:48.428+0530 I CONTROL  [initandlisten]
2016-01-06T18:00:49.745+0530 I NETWORK  [HostnameCanonicalizationWorker] Startin
g hostname canonicalization worker
2016-01-06T18:00:49.746+0530 I FTDC     [initandlisten] Initializing full-time d
iagnostic data capture with directory 'c:/mongodbsetup/data/diagnostic.data'
2016-01-06T18:00:49.769+0530 I NETWORK  [initandlisten] waiting for connections
on port 27017
```

Fig -2: Starting MongoDB

Fig. 2 shows starting MongoDB, run mongod.exe. This starts the main MongoDB database process. The waiting for connections message in the console output indicates that the mongod.exe process is running successfully.



Enter time span :

Process completed

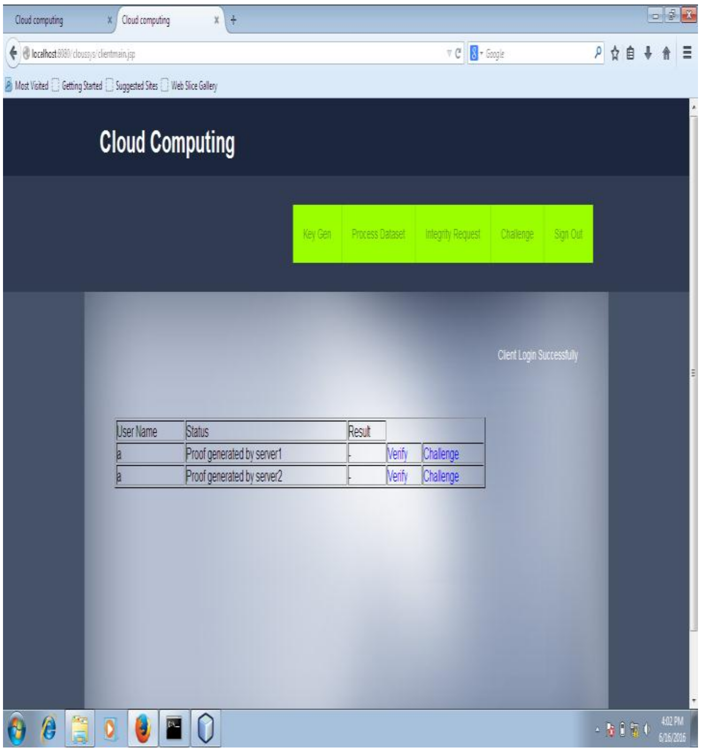
Keys

LC1nDxlxMIWyNPDUbDcHk0FxyNgjmL0t0hx1PxQzVF0tO/jgmybqHQmlUEyEKW16vcroEJ3tDrvh2E
/9KpolMMeqhi6ThTwJ+xzvaMATqKUFM2lhKYTfrkUGfv2slbbN17J6RDuAAgPySFAqXqTsLaG7NF+pstVv

Xdd6a+EduydMAjMvXwyiq08znM9c40V52R6fQ6yyb3t1E7AoSmPNHmPCbx7+MCkY9oBj4zQ7kU+5x+JQVCl5uldIUXFVWU18
vpoGxF/76hMdGe8TSwX3ZP6gHX/zhbrnDS2EK0apuJpvdrYqBw8udwtuts4nQ7gm

Fig -3: Key Generation and Updation

Fig. 3 shows: The client generates the set of secret key and public key by using key generation and key updation algorithms. The application accepts the time span from the client and creates the entered number of secret keys which will be assigns automatically to TPA periodically to strengthen the security.



Client Login Successfully

User Name	Status	Result		
a	Proof generated by server1	-	Verify	Challenge
a	Proof generated by server2	-	Verify	Challenge

Fig - 4: Challenge-Proof-Verify Process

Fig. 4 shows Integrity verification by user to check proof generated by TPA to check trustworthiness of TPA.

5. RESULT AND DISSCUTION ON MODULES

Table-1: Tag generation time:

DATASET (MB)	TIME(MS)
1	305
2	415
3	503
4	610
5	708

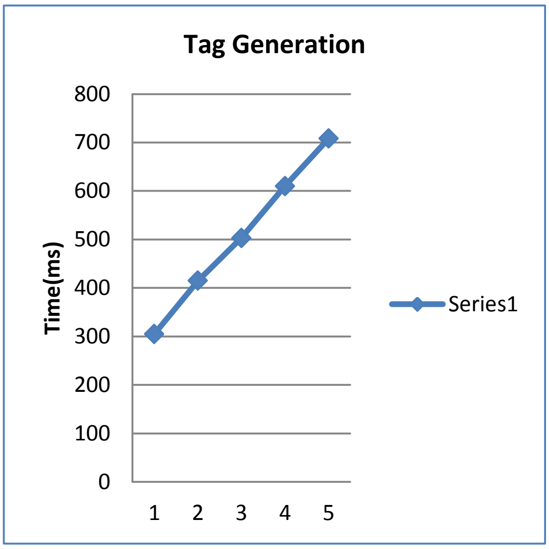


Chart -1: Tag generation time on Server

In the first experiment, Authenticator generation time is calculated. Five different number of dataset sizes are chosen in the experiment to show the authenticator generation. From chart 1, we can see that the time needed to generates authenticator's increases when the size of dataset increases. Compared with the previous related work [1] time needed to this is less.

Table -2: Challenge-Proof-Verify time on server 1

Dataset(MB)	Time (ms)
1	45
2	60
3	62
4	72
5	75

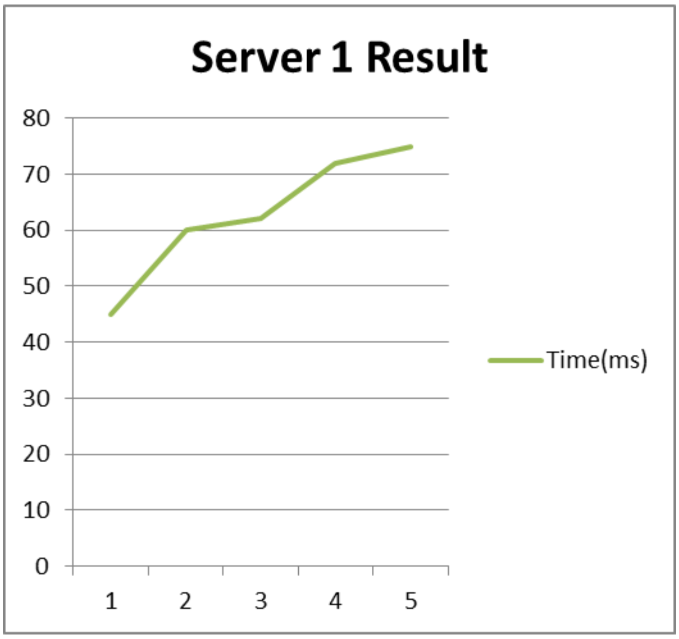


Chart - 2: Time required for Challenge-Proof-Verify process at Server 1(For first Replica of data file)

Table -3: Challenge-Proof-Verify time on server 2

DATASET (MB)	TIME(MS)
1	61
2	72
3	95
4	105
5	120

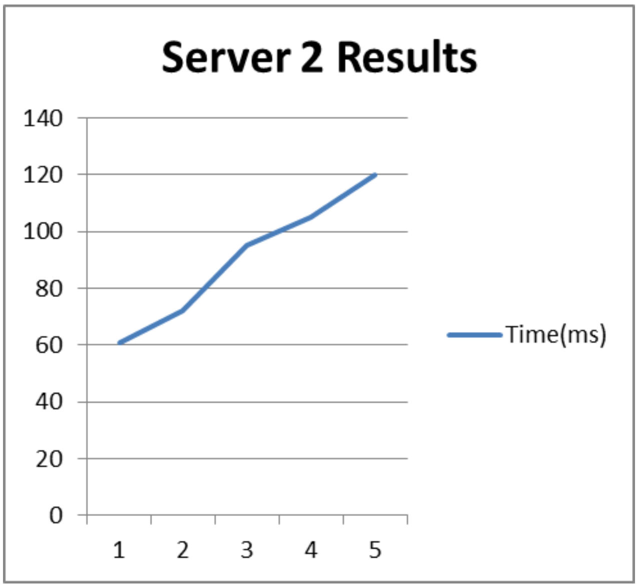


Chart - 3: Time required for Challenge-Proof-Verify process at Server 2(For second Replica of data file)

In the second experiment, we have calculated the time required to for Challenge-Proof-Verify process at different servers for multiple replicas of data file using No-SQL database.

From the charts 2 and 3, we can say that time needed for Challenge-Proof-Verify process at different servers for multiple replicas of data file using No-SQL database is quite less than traditional RDBMS.

6. CONCLUSIONS

This paper proposes a new proof of retrievability for cloud storage, this proposes a protocol that verifies CSS's as well as TPA's generated integrity proofs for multiple replica of a data file. This paper replaces traditional relational database management system by No-SQL database. It provides Decentralized access to data, maintains replicas of dataset which produces availability, data scalability and stores large volume of data.

REFERENCES

[1] Jin Li, Xiao Tan, Xiaofeng Chen, Duncan S. Wong, and Fatos Xhafa, "OPoR: Enabling Proof of Retrievability in Cloud Computing with Resource-Constrained Devices" IEEE Transactions on cloud computing vol.3, No.2, April/June 2015.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson,and D. Song, "Provable data possession at untrusted stores,"in Proc. 14th ACM Conf. Comput. Commun. Security, 2007, pp. 598–609.

[3] A. Juels and B. S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in Proc. 14th ACM Conf. Comput. Commun. Security, 2007, pp. 584–597.

[4] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. 14th Int. Conf. Theory Appl. Cryptol. Inf. Security, 2008, pp. 90–107.

[5] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability:Theory and implementation," in Proc. ACM Workshop Cloud Comput.Security, 2009, pp. 43–54.

[6] M. Naor and G. N. Rothblum, "The complexity of online memory checking," J. ACM, vol. 56, no. 1, pp. 2:1–2:46, Feb. 2009.

[7] E.-C. Chang and J. Xu, "Remote integrity check with dishonest storage server," in Proc. 13th Eur. Symp. Res. Comput. Security,2008, pp. 223–237.

[8] M. A. Shah, R. Swaminathan, and M. Baker. (2008). Privacy-preserving audit and extraction of digital contents, Cryptology ePrint Archive, Report 2008/186 [Online]. Available: http://eprint.iacr.org/

[9] A. Opera, M. K. Reiter, and K. Yang, "Space-efficient block storage integrity," in Proc. 12th Annu. Netw. Distrib. Syst. Security Symp. 2005, pp. 17–28.

[10] T. S. J. Schwarz and E. L. Miller, “Store, forget, and check: Using algebraic signatures to check remotely administered storage,” in Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst., 2006, p. 12.

[11] Q. Wang, K. Ren, S. Yu, and W. Lou, “Dependable and secure sensor data storage with dynamic integrity assurance,” ACM Trans.Sens. Netw., vol. 8, no. 1, pp. 9:1–9:24, Aug. 2011.

[12] Jia Yu, Kui Ren, Cong Wang and Vijay Varadharajan, “Enabling Cloud Storage Auditing with Key-Exposure Resistance” IEEE Transactions on Information forensics and security vol.10, No.6, June 2015.

[13] Zhuo Haoand Nenghai Yu, “A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability” IEEE Transactions on Data, Privacy and E-commerce April 2010.

[14] Ruxandra Burtica, Eleonora Maria Mocanu, Mugurel Ionuţ Andreica, Nicolae Ţapuş “Practical application and evaluation of No-Sql databases inCloud Computing” IEEE transactions on No-Sql 2012.

[15] Jing Han, Haihong E, Guan Le and Jian Du ” Survey on No-Sql Database” IEEE 2011.

[16] MongoDB tutorials https://docs.mongodb.org/getting-started/shell/import-data/

[17] MongoDB predefined datasets https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/dataset.json

[18] R.Hecht and S.Jablonski, “NoSQL evaluation: A use case oriented survey. In Cloud and Service Computing (CSC) ”, 2011 International Conference, IEEE, (2011) December, pp. 336-341.

[19] Use relational DBMS, N. saying good bye to DBMSs, designing effective interfaces, Communications of the ACM vol.52 no. 9, (2009)

Mr. Mahesh Pore received the B.E. degree in Computer Science in 2007 from Shivaji University. He is currently working as Lecturer in Institute of Civil and Rural Engineering polytechnic, Gargoti (Maharashtra, India). His research interest includes Cloud Computing and Security.

Mr. Sachin Takmare received the M.Tech. Degree in Computer Science and Engineering in year 2013 from Rajiv Gandhi Technical university, Madhya Pradesh. He is currently working as Head of the Department ,Computer Engineering, Bharati Vidyapeeth’s College of Engineering, Kolhapur, and Maharashtra, India. His research interest includes Data Mining, Networking and Image Processing.

Mr. Amol Mahadik received the M.E. degree in Computer Engineering in year 2015 from Savitribai Phule University Pune, Maharashtra. He is currently working as Lecturer, Computer Engineering,Bharati Vidyapeeth’s Institute of Technology, Palus, and Maharashtra, India. His research interest includes Data Mining, Cloud Computing and Big Data.