ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 1, January 2013

# Survey on Data Integrity in Cloud Computing

**Reenu Sara George, Sabitha S**

***Abstract*— Data outsourcing to cloud storage has recently become a trend. Economic advantage is the significant reason. A large number of clients or data owners store their important data in remote servers in the cloud and it is provided back to them whenever needed. Data owners don't even leave a copy of the data in their local computers so as to save the storage space. By sending the data to be processed in the cloud, data owners transfer control of their data to a remote party that may raise security challenges. Sometimes the data stored in the cloud is so important that the clients must ensure that it is not lost or corrupted, i.e., the integrity of data has to be ensured. A lot of works have been done on designing remote data integrity checking protocols, which allow data integrity to be checked without completely downloading the data. This paper surveys the various protocols to check cloud data integrity and compares them based on the integrity requirements.**

*Index Terms*— **Integrity, Cloud computing, Public verifiability, Privacy preservation, Data possession**

## I. INTRODUCTION

Cloud Computing is a new trend in the field of Information Technology. More and more users store their data in clouds that can be accessed remotely over the Internet. Even though the advantages of cloud computing are immense, there are a lot of security challenges too. One important security problem is guaranteeing the integrity of remotely stored data. In computer security, data integrity can be defined as "the state that exists when computerised data is the same as that in the source documents and has not been exposed to accidental or malicious alteration or destruction". Simply, integrity means preventing unauthorised modification of data. It includes both intentional modification such as insertion or deletion of malicious data and unintentional modification such as random transmission error.

Integrity of data stored at the untrusted cloud server is not guaranteed. For example, the cloud service providers may decide to hide the data errors from the client for the business profit. They may delete the data that are rarely accessed by the client. Thus client should need knowledge about the remote data by continuously monitoring the integrity of the stored data.

In a remote data integrity checking protocol, the data owner (client) initially stores data and metadata in the cloud storage (server); later, an auditor (the data owner or another client) can challenge the server to prove that it can produce the data that was originally stored by the client; the server then generates a proof of data possession based on the stored data and metadata. A lot of works have been done on designing remote data integrity checking protocols, which allow data integrity to be checked without completely downloading the data. This paper surveys the various protocols to check cloud data integrity. These protocols are analysed and compared based on the integrity requirements.

***Manuscript received Jan, 2013.***

***Reenu Sara George,*** *Department of Computer Science and Engineering, College of Engineering, Trivandrum.*

***Sabitha S****, Department of Computer Science and Engineering, College of Engineering, Trivandrum.*

## II. BACKGROUND

### A. Basic Concepts

Fig. 1 shows the cloud storage architecture. Three main entities in cloud environment include:

- Cloud storage server: This is managed by the cloud service provider (CSP). It provides data storage service and has significant resources.
- User: Users own the data to be stored in the cloud and access them when needed. They rely on the cloud for data computation.
- Third party auditor: An optional TPA is trusted to assess and expose risk of cloud storage services on behalf of the user's open request.

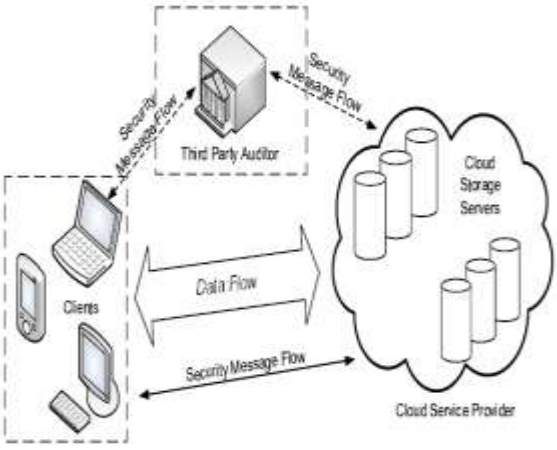


Fig. 1: Cloud Storage Architecture

### B. Characteristics of remote data integrity checking protocols

While designing a remote data integrity checking protocol, certain requirements have to be satisfied:

- Privacy preservation: The TPA should not gain knowledge of the original user data during the auditing process.

- Unbound number of queries: The verifier may be allowed to use unlimited number of queries in the challenge-response protocol for data verification.
- Data dynamics: The clients must be able to perform operations on data files like insert, alter and delete while maintaining data correctness.
- Public verifiability: Anyone, not just the clients, must be allowed to verify the integrity of data.
- Blockless verification: Challenged file blocks should not be retrieved by the verifier during verification process.
- Recoverability: Apart from checking correct possession of data, some scheme to recover the lost or corrupted data is required.

## III. Integrity Checking Protocols

In [1], a Provable Data Possession (PDP) model which is provably secure for remote data checking is constructed. They introduced the concept of RSA-based homomorphic verifiability tags (HVTs) which are the building blocks for PDP scheme. A PDP protocol checks whether an outsourced storage site retains a file, which consists of a collection of data blocks. The client pre-processes the file, generating a piece of metadata that is stored locally, transmits the file along with metadata to the server and may delete its local copy. The server stores the file and responds to challenges issued by the client. As part of pre-processing, the client may alter the file to be stored at the server as shown in fig. 2. The client may expand the file or include additional metadata to be stored at the server. Before deleting its local copy of the file, the client may execute a data possession challenge to make sure the server has successfully stored the file. Clients may encrypt a file prior to out-sourcing the storage. At a later time, the client issues a challenge to the server to establish that the server has retained the file. The server computes a proof of possession, which it sends back to the client as depicted in fig.3. Using its local metadata, the client verifies the response. This scheme:

- allows public verifiability
- lack of privacy preservation
- no dynamic support
- unbound no. of queries

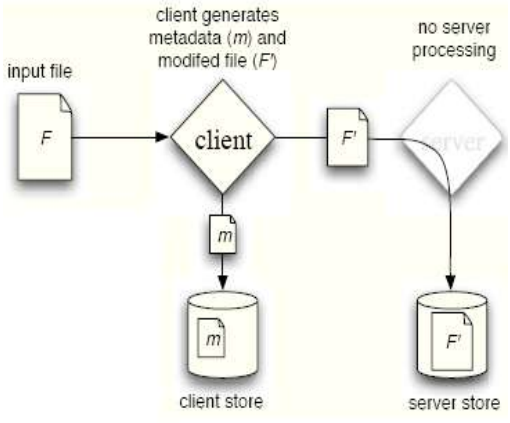


Fig 2: Pre-process and store

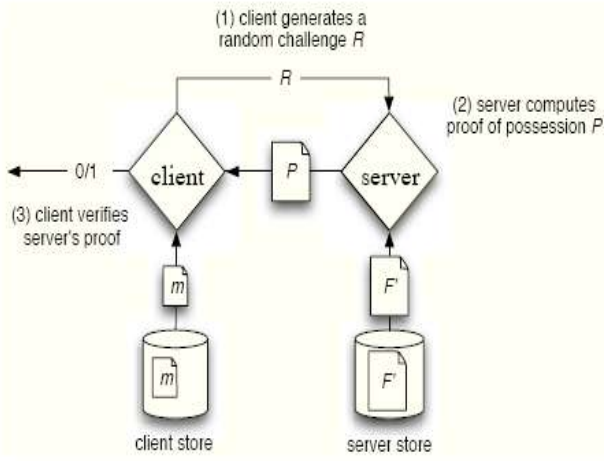


Fig. 3: Verify data possession

In [2], Proofs of Retrievability (POR) which uses only a single cryptographic key is proposed. Also the archive needs to access only a small portion of the file. POR protocol encrypts the file and embeds a set of randomly-valued check blocks called sentinels. The use of encryption here renders the sentinels indistinguishable from other file blocks. The verifier (client) challenges the prover(server) by specifying the positions of a collection of sentinels and asking the prover to return the associated sentinel values. If the prover has modified or deleted a substantial portion of file, then with high probability it will also have suppressed a number of sentinels. It is therefore unlikely to respond correctly to the verifier. This is best suited for storing encrypted files. This scheme:

- prevents dynamic auditing
- support only a limited number of queries
- recoverability
- privacy preserving

Scheme [3] is a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption. This scheme is more efficient than POR as it requires no bulk encryption of outsourced data and no data expansion due to additional sentinel blocks. It is based entirely on symmetric-key cryptography. The main idea is that, before outsourcing, owner pre-computes a certain number of short possession verification tokens, each token covering some set of data blocks. The actual data is then handed over to server. Subsequently, when owner wants to obtain a proof of data possession, it challenges server with a set of random-looking block indices. In turn, server must compute a short integrity check over the specified blocks and return it to owner. For the proof to hold, the returned integrity check must match the corresponding value pre-computed by owner. However, in this scheme owner has the choice of either keeping the pre-computed tokens locally or outsourcing them (in encrypted form) to server.

This scheme:

- supports dynamic data operations like update, delete and append
- not fully dynamic as it does not support insertion
- the prefixed number of verifications
- no public verifiability

In [4], a new scheme is proposed which allows a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. To achieve efficient data dynamics, the proposed scheme improves the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication. A Merkle Hash Tree (MHT) is a well-studied authentication structure, which is intended to efficiently and securely prove that a set of elements are undamaged and unaltered. It is constructed as a binary tree where the leaves in the MHT are the hashes of authentic data values. To support efficient handling of multiple auditing tasks, it used the technique of bilinear aggregate signature to extend it into a multiuser setting, where TPA can perform multiple auditing tasks simultaneously. Features of this scheme include:

- public auditability
- fully dynamic data operation
- blockless verification
- no privacy preservation

In [5], a new remote integrity checking protocol based on homomorphic verifiable tags are discussed. The proposed protocol has the functions SetUp, TagGen, Challenge, GenProof and CheckProof, as well as functions for data dynamics. SetUp generates a pair of keys, one publicly known to everyone, and other kept secret by the client. Using TagGen, client computes the block tag for each file block. Let $D_m$ be the set of block tags. After finishing computing all the block tags, the client sends the file *m* to the remote server, and releases $D_m$ to be publicly known to everyone. In order to verify the integrity of the file *m*, the verifier generates a challenge using Challenge function and sends it to the server. When the server receives the challenge, it executes GenProof to compute the proof and sends it to the verifier. When the verifier receives the response from the server, it checks its validity. If it is valid, the function outputs "success," otherwise the function outputs "failure." Main features of this scheme include:

- security against the server with public verifiability
- privacy against third-party verifiers
- data dynamics at block level

## IV. Comparative Study

The remote data integrity checking protocols are well analyzed and a comparative study is done based on whether their requirements are satisfied or not. Table I shows the result.

TABLE I
Comparative Study of Various Protocols

| | [1] | [2] | [3] | [4] | [5] |
|---|---|---|---|---|---|
| **Privacy Preservation** | No | Yes | No | No | Yes |
| **Unbound number of queries** | Yes | No | No | Yes | Yes |
| **Data dynamics** | No | No | Yes(not fully dynamic) | Yes | Yes |
| **Public verifiability** | Yes | No | No | Yes | Yes |
| **Blockless verification** | Yes | Yes | Yes | Yes | Yes |
| **Recoverability** | No | Yes | No | No | No |
| **With the help of TPA** | No | No | No | Yes | No |
| **Untrusted server** | Yes | Yes | Yes | Yes | Yes |

## V. Conclusion

In this paper, the various protocols to check remote data integrity in cloud computing were studied, analyzed and compared. Most of the previous works deal only with static data while recent works support dynamic operations. Some schemes allow public verifiability while others not. All schemes deal with integrity in untrusted servers. Newly introduced protocols are designed such a way that they support public auditability and preserves privacy of data. Designing efficient, secure and fully dynamic remote data integrity verification protocol scheme with recoverability feature is still a hot topic in research.

## References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stores", *Proceedings of the 14th ACM conference on Computer and communications security*, CCS 07 ,New York, USA, ACM, 2007, pp. 598-609.

[2] Juels A, Kaliski BS, "PORs: proofs of retrievability for large files", *Cryptology ePrint archive*, June 2007, Report 2007/243.

[3] G. Ateniese, et al., "Scalable and efficient provable data possession", *Proceedings of the 4th international conference on Security and privacy in Communication networks*, Istanbul, Turkey, 2008.

[4] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 5, May 2011

[5] Z. Hao, S. Zhong and N. Yu,"A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 23, No. 9, September 2011

[6] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing", *Proc. 17th Int'l Workshop Quality of Service (IWQoS '09)*, pp. 1-9, July 2009.

[7] H. Shacham and B. Waters, "Compact proofs of retrievability", *Proc. of ASIACRYPT' 08.* Melbourne, Australia: Springer-Verlag, 2008, pp. 90-107.