# CHAPTER -1

# INTRODUCTION

## Cloud computing is foreseen as the next generation of the information technology architecture (IT) for enterprises, due to its long list of unprecedented advantages in the IT history[1].

The main intention of cloud computing is to transform the nature of how businesses use information technology. One fundamental aspect of this paradigm shifting is that the data needs to be centralized or outsourced to the cloud. While outsourcing the data to the cloud, also brings the new and challenging security (TPA) threats toward users’ outsourced data [4]. In this report the auditing of the third party by preserving the privacy of the data which is stored in the cloud, is discussed by the following ways:-

1. To enable the external auditor to audit user’s cloud data without learning the data content.
2. To find a way in which the multiple auditors can perform auditing simultaneously in a privacy-preserving manner.
3. To prove the security and justify the performance of schemes.

##### 1.1 What does it mean to preserve privacy?

Assume that we have some kind of data set that is privacy relevant, it includes or is based on potentially sensitive information about individuals.  We want to let an analyst get answers to questions about the data in aggregate, but without letting analyst deduce private information about any individual [1].

Defining privacy-preserving data access turns out to be harder than you might expect, because many superficially attractive definitions break down [5].  Some definitions work if the analyst can ask only one question, but get into trouble if the analyst can ask multiple questions–the analyst might be able to ask two “harmless” questions whose answers, taken together, reveal everything about an individual.   Some definitions break down if the analyst has any access to “outside” information about the world, even basic information like the fact that the average human is less than ten feet tall.  Some definitions turn out to be impossible to achieve in this universe. Differential privacy avoids all of these pitfalls.

The core of the definition is simple but subtle. Imagine two data sets, A and B, which are exactly identical except that A includes information about one additional person who is not included in B. Now we ask: If we let the analyst get answers to questions about our data set, can the analyst tell whether we are using data set A or data set B? If the analyst can’t tell the difference and this result holds no matter which data set A we started with and no matter which person we excluded from B, then we have succeeded in preserving privacy. Why?  Because any inference the analyst makes about you will have to be the same inference he would have made even if your information were not in the data set at all. To put it another way, including your information in the data set did nothing to harm your privacy.

Now there is one more itty-bitty problem, which is that the definition as stated so far can’t quite be achieved, so we have to modify the definition slightly. Instead of saying that the analyst’s results have to be exactly identical whether we’re using data set A or B, we will say that the results have to be almost identical. We will allow a very small probability that the analyst’s answers depend on whether we are using A or B.   For example, if the probability that the analyst can tell the difference is less than 0.01%, then we’ll say that we have achieved differential privacy at the 0.01% level.

##### 1.2 PRIVACY ISSUES IN CLOUD COMPUTING

Privacy [7] could be understood as the right of a person to have his personal data properly secured. Any data that could uniquely identify a person or, which is not supposed to be known to any person other than its owner and/or her immediate family, without her consent is called Private Data. It is therefore needed to maintain the confidentiality of private data. However, current cloud services usually cause this data to be exposed, on a machine owned and operated by an organization which is different from the data owner, in unencrypted form. The major privacy issues in cloud computing environment relate to:

a. Trust

b. Uncertainty

c. Compliance

The most common issues related to privacy in cloud computing environment are:

* Lack of User Control:

Complete control of user on the data is not possible in the cloud, since both the visibility and control of a user is reduced as soon as the cloud environment is used.

* Lack of Training and Expertise:

The deployment and running of cloud services requires high skill jobs, but the unavailability of highly skilled people is a serious issue from the point of view of information security.

* Unauthorized Secondary Usage:

The risk of unauthorized use of data either stored or processed in the cloud is always present. The authorized secondary usage of any user’s data by the service provider to gain revenue is part of standard business model. However, the data could also be used in a way which is unacceptable to the user. Therefore, it is necessary for the cloud service providers and the customers to enter into a legally binding agreement which explicitly mentions as to how and up to what extent the data of a customer could be used. This will also enhance the trust between the customer and the cloud service provider.

* Complexity of Regulatory Compliance:

It is difficult to even know exactly where the data is or if it is in transit at any given time. Another factor which complicates the compliance issue is the presence of multiple copies of same data in the cloud, and each of these copies may be managed by different entities. The main properties which make compliance difficult are:

1. Data Proliferation: This is the feature of cloud computing in which, to ensure the availability of some data, the cloud providers replicate that data in multiple locations.
2. Dynamic Provisioning: The problems related to outsourcing which cloud computing environment faces are quite similar to that in traditional outsourcing, but the dynamic nature of cloud computing environment makes many of the existing provisions which address these issues in static environment obsolete.

* Trans border Data Flow:

Privacy laws and data protection regulations restrict the flow of private data outside the national borders, restricting not only the physical transfer of data but also remote access to the data. All countries having national legislations have restricted such transfers.

* Litigation:

A government may force a cloud service provider to give them the data stored in the cloud. All they have to do is to show that the requested data is relevant to some case for a subpoena.

* Legal Uncertainty:

Legal frameworks have played very important role in the protection of the personal and sensitive information of any user. The basic concepts of such legal frameworks are generally technology neutral, and therefore they would still be applicable on cloud computing environment. Still these frameworks need to be updated keeping the current and future technologies in consideration.

## 

## CHAPTER 2

## LITERATURE SURVEY

**2.1 A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability** [**Zhuo Hao etl. 2011]**

The paper proposes method called remote integrity checking [1] with following issues

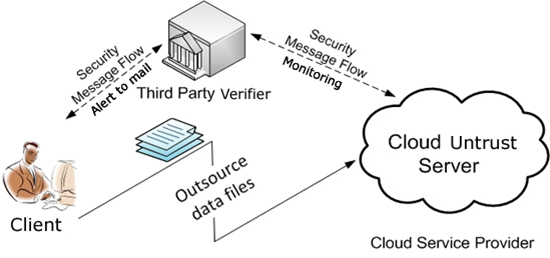
**Data dynamics**: means after clients store their data at the remote server, they can dynamically update their data at later times.

**Public verifiability**: allows anyone (not just the client) to perform the integrity checking operation.

**Privacy against verifiers**: When the verification is performed by a third party verifier (not by the client), the protocol must ensure that no private information contained in the data is leaked.

* **Architectural diagram:**

The architectural diagram used is shown below. Where there are three main components client, third party verifier and cloud untrusted server. The client outsourced data files to the cloud server. Third party verifier monitors for any changes that occurred in cloud server. Also it alerts the client by using security messages.



**Fig 2.1.1: Architectural Diagram**

* **Basic algorithms:**

Denote by m the file that will be stored in the untrusted server, which is divided into n blocks of equal lengths: m = m1m2...mn, where n = ⌈|m|/l˥. Here l is the length of each file block. Denote by fK(·) a

Pseudo-random function which is defined as:

f : {0, 1}k × {0, 1}log2(n) → {0, 1}d

In which k and d are two security parameters. Furthermore, denote the length of N in bits by |N|.

To design remote data integrity checking protocol that includes the following five functions: SetUp, TagGen, Challenge, GenProof and CheckProof.

**SetUp(1k) → (pk, sk):** Given the security parameter k, this function generates the public key pk and the secret key sk. pk is public to everyone, while sk is kept secret by the client.

**TagGen(pk, sk,m) → Dm:** Given pk, sk and m, this function computes a verification tag Dm and makes it publicly known to everyone. This tag will be used for public verification of data integrity.

**Challenge (pk,Dm) → chal:** Using this function, the verifier generates a challenge chal to request for the integrity proof of file m. The verifier sends chal to the server.

**GenProof (pk,Dm, m, chal) → R:** Using this function, the server computes a response R to the challenge chal. The server sends R back to the verifier.

**CheckProof (pk,Dm, chal,R) → {“success”, “failure”}:** The verifier checks the validity of the response R. If it is valid, the function outputs “success”, otherwise the function outputs “failure”. The secret key sk is not needed in the Check Proof function.

Besides these five functions, if the protocol supports data dynamics, it should also have functions such as block insertion, block deletion, and block modification.

* **Protocol used:**

The remote data integrity checking protocol [1] uses homomorphic verifiable tags (HVT) . A HVT of a message m is a pre-computed tag which is later used for the integrity checking. Denote the HVT of a message mi by Di. The HVT has the following two features:

• Block less verification: By using HVTs, the server can construct a proof of possession of a certain file blocks, while the client needs not have access to these file blocks.

*•* Homomorphic property: For any two messages *mi* and *mj* , the tag for *mi*+*mj* can be generated by combining *Di* and *Dj* .

The author uses a RSA-based HVT, which is defined as follows. Let *N* = *pq* be one publicly known RSA modulus.

Theorems proposed in this paper:

Theorem 1: If both the client and the server are honest, then the server can pass the verification successfully.

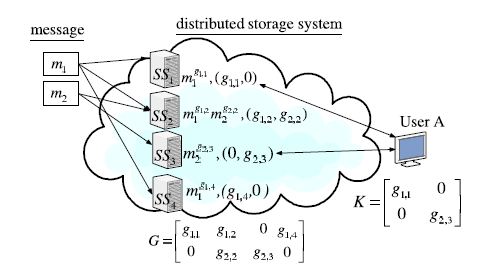
Theorem 2: (Privacy against Third Party Verifiers) Under the semi-honest model, a third party verifier cannot get any information about the client’s data m from the protocol execution. Hence, the protocol is private against third party verifiers.

* **Privacy Against Third Party Verifier:**

Under the semi-honest model, a third party verifier cannot get any information about the client’s data m from the system execution. Hence, the system is private against third party verifiers. If the server modifies any part of the client’s data, the client should be able to detect it; furthermore, any third Party verifier should also be able to detect it. In case a third party verifier verifies the integrity of the client’s data, the data should be kept private against the third party verifier.

* 1. **An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing [Kan Yang etl. 2012**]

In the proxy Re-encryption key the messages are first encrypted by the owner and then stored in a storage server [2]. When a user wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the encrypted messages for the authorized user. Thus, their system has data confidentiality and supports the data forwarding function.



**Fig 2.2.1 A storage system with random linear coding over exponents**

An encryption scheme is multiplicative homomorphic if it supports a group operation on encrypted plaintexts without decryption. The multiplicative homomorphic encryption scheme supports the encoding operation over encrypted messages. Then convert a proxy re-encryption scheme with multiplicative homomorphic property into a threshold version. A secret key is shared to key servers with a threshold value t. To decrypt for a set of k message symbols, each key server independently queries 2 storage servers and partially decrypts two encrypted codeword symbols. As long as t key servers are available, k codeword symbols are obtained from the partially decrypted cipher texts.

In order to preserve privacy [2], the clients will encrypt their data when they out- source it to the cloud. However, the encrypted form of data greatly impedes the utilization due to its randomness. Many efforts have been done for the purpose of data usage but without undermining the data privacy.

Homomorphism: Given two cipher texts c1 and c2 on plaintexts m1 and m2 respectively, one can obtain the cipher text on the plaintext m1 +m2 and/or m1 ·m2 by evaluating c1 and c2 without decrypting cipher texts.

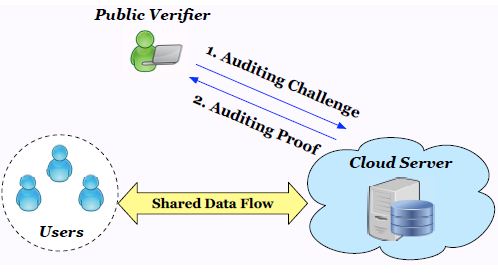
Proxy re-encryption: Given a proxy re-encryption key, the proxy can transform a cipher text of one user to a cipher text of the target user.

Threshold decryption: By dividing the private key into several pieces of secret shares, all clients can work together to decrypt the cipher text – the output of the function.

* 1. **Oruta : Privacy-Preserving Public Auditing for Shared Data in the Cloud**

**[B. Wang etl. 2014 ]**

The following architecture is used in Oruta [3]. Where there are group of users, public verifier and cloud server.



**Fig 2.3.1: Architecture used in Oruta**

The concept of ring signatures was first proposed by Rivest et al. in 2001. With ring signatures, a verifier is convinced that a signature is computed using one of group members’ private keys, but the verifier is not able to determine which one. More concretely, given a ring signature and a group of d users, a verifier cannot distinguish the signer’s identity with a probability more than 1/d. This property can be used to preserve the identity of the signer from a verifier. The ring signature scheme introduced by Boneh et al. (referred to as BGLS in this paper) is constructed on bilinear maps. Author will extend this ring signature scheme to construct our public auditing mechanism.

Homomorphic authenticators (also called homomorphic verifiable tags) are basic tools to construct public auditing mechanisms. Besides unforgeability (i.e., only a user with a private key can generate valid signatures), a homomorphic authenticable signature scheme, which denotes a homomorphic authenticator based on signatures, should also satisfy the following properties:

Block less verifiability

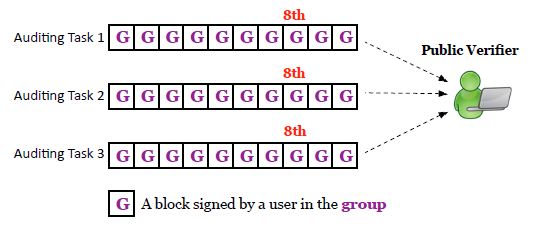
Non-malleability

Author proposed a new homomorphic authenticable ring signature (HARS) scheme, which is extended from a classic ring signature scheme. The ring signatures generated by HARS are not only able to preserve identity privacy but also able to support blockless verifiability. HARS contains three algorithms: **KeyGen**, **RingSign** and **RingVerify**.

In **KeyGen**, each user in the group generates his/her public key and private key. In **RingSign**, a user in the group is able to generate a signature on a block and its block identifier with his/her private key and all the group members’ public keys. A block identifier is a string that can distinguish the corresponding block from others. A verifier is able to check whether a given block is signed by a group member in **RingVerify**.

Oruta, a privacy preserving public auditing mechanism for shared data in the cloud. With Oruta, the public verifier can verify the integrity of shared data without retrieving the entire data. Meanwhile, the identity of the signer on each block in shared data is kept private from the public verifier during the auditing.

According to the generation of ring signatures in HARS, a block m is an element of Zp and its ring signature contains d elements of G1, where G1 is a cyclic group with order p. It means a |p|-bit block requires a d×|p|-bit ring signature, which forces users to spend a huge amount of space on storing ring signatures.



**Fig 2.3.2 Alice and Bob share a file in the cloud, and a public verifier audits the integrity**

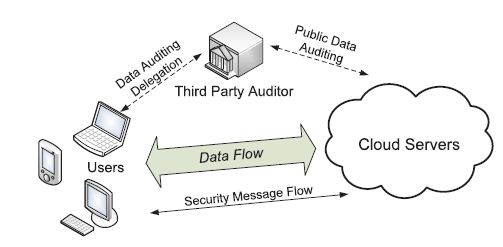
**2.4. Privacy preserving public auditing for secure cloud storage**

**[C. Wang etl. 2013]**

author proposed to uniquely integrate the homomorphic linear authenticator[4] with random masking technique.

* **Architecture:**

The proposed architecture or the privacy preserving public auditing cloud data storage is as shown below:



**Fig 2.4.1: the architecture of cloud data storage**

* **Algorithm:**

A public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof).

* KeyGen: key generation algorithm that is run by the user to setup the scheme
* SigGen: used by the user to generate verification metadata, which may consist of MAC, signatures or other information used for auditing
* GenProof: run by the cloud server to generate a proof of data storage correctness
* VerifyProof: run by the TPA to audit the proof from the cloud server

1. Key Generation: The owner generates a public/secret key pair (pk, sk) by himself or the system manager, and then sends his public key pk to TPA. Note that TPA cannot obtain the client’s secret key sk; secondly, the owner chooses the random secret.

2. Tag Generation: The client (data owner) uses the secret key sk to pre-process a file, which consists of a collection of n blocks, generates a set of public verification parameters and index-hash table that are stored in TPA, and transmits the file and some verification tags to CSP.

3. Periodic Sampling Batch Audit: The Batch TPA (or other applications) issues a “Random Sampling” challenge to audit the integrity and availability of outsourced data in terms of the verification information stored in TPA.

4. Audit for Dynamic Operations: An authorized application, which holds data owner’s secret key sk, can manipulate the outsourced data and update the associated index hash table stored in TPA. The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the authorized applications and forge the valid audit records.

* **Privacy-Preserving Public Auditing Module**:

Homomorphic authenticators are unforgeable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an auditor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. Overview to achieve privacy-preserving public auditing, author propose to uniquely integrate the homomorphic authenticator with random mask technique. In our protocol, the linear combination of sampled blocks in the server’s response is masked with randomness generated by a pseudo random function (PRF).

The proposed scheme is as follows:

* + - * Setup Phase
      * Audit Phase

1. Setup phase: The user initializes the public and secret parameters of the system by executing KeyGen, and preprocesses the data file F by using SigGen to generate the verification metadata. The user then stores the data file F and the verification metadata at the cloud server, and delete its local copy. As part of preprocessing, the user may alter the data file F by expanding it or including additional metadata to be stored at server.
2. Audit phase: The TPA issues an audit message or challenge to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will derive a response message by executing GenProof using F and its verification metadata as inputs. The TPA then verifies the response via VerifyProof.

**Fig 2.4.2: setup and audit phase**

* **Batch Auditing Module**:

With the establishment of privacy-preserving public auditing in Cloud Computing, TPA may concurrently handle multiple auditing delegations upon different users’ requests. The individual auditing of these tasks for TPA can be tedious and very inefficient. Batch auditing not only allows TPA to perform the multiple auditing tasks simultaneously, but also greatly reduces the computation cost on the TPA side.

* **Data Dynamics Module:**

Supporting data dynamics for privacy-preserving public risk auditing is also of paramount importance. How the main scheme can be adapted to build upon the existing work to support data dynamics, including block level operations of modification, deletion and insertion. Author adopts this technique in design to achieve privacy-preserving public risk auditing with support of data dynamics.

**CHAPTER 3**

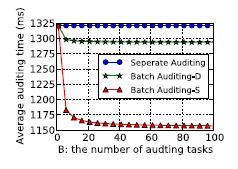
**COMPARISON & ANALYSIS**

As there are many methods we have studies from the different authors, each method have some advantage and some disadvantage. The following table shows comparison between the methods we discussed in previous chapter.

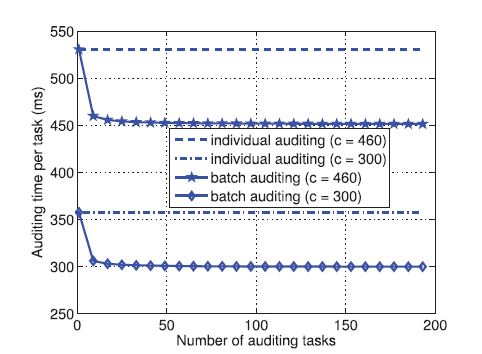
Table: comparison between different mechanisms used in the papers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Paper1[1] | Paper2[2] | Paper3[3] | Paper4[4] |
| Methodology used | Remote data integrity protocol | Threshold proxy re-encryption scheme | Ring signature scheme | Homomorphic linear  Authenticator with random masking technique |
| Random masking technique | Does not support | Does not support | Supports | Supports |
| Public verifiability | Without help of a third party | With the help of a third party | With the help of a third party | With the help of a third party |
| Data privacy | Supported | Supported | Supported | Supported |
| Identity privacy | Not implemented | Not implemented | Not implemented | Implemented |
| Auditing | Remote storage auditing | There is no auditing mechanism | Batch auditing supported | Batch auditing supported |
| Data dynamics | Supported | Not supported | Supported | Supported |
| TPA to perform auditing | There is no TPA | There is no auditing mechanism | With minimum communication and computation  Overhead | With minimum communication and computation  Overhead |

The following figures show the comparison between the batch auditing.

.

**Fig 3.1** Performance Batch Auditing in paper 3

****

**Fig 3.2** Performance Batch Auditing for paper 4

From the above two figures it is clear that the performance of the paper 4 is much better than the batch auditing mechanism explained in paper 3.

**CHAPTER 4**

**CHALLENGES & FUTURE DIRECTIONS**

In the future work, we hope to build an efficient public auditing mechanism with the capability of preserving identity privacy and public auditing mechanism using TPA. Also we will try to introduce a secure scheme for data dynamics in such a way that for performing any transaction, client admin is not required to decrypt the records so that it saves time and the cost of bandwidth.

In order to provide client full advantage of acquiring a cloud storage service, client admin will store the VMd, private and public keys with TPA, not at their local machine, however these parameters are encoded and can be decrypted only by the privileged user.

Furthermore, we hope to optimize the system architecture and modules proposed in this paper.

**CHAPER 5**

**CONCLUSION**

We have studied various privacy issues such as data dynamics solutions, and batch auditing schemes with data integrity solutions.

In this report we discussed various data privacy preserving solution for organizations which deals with sensitive information where the TPA would not have any knowledge about the data content stored on the cloud server during the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the user’s fear of their outsourced data being leakage.

We pointed out the issues in privacy preserving mechanism which we will work upon in the future.

**REFRENCES**

[1] Zhuo Hao, Sheng Zhong, Nenghai Yu, “A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability”, IEEE Transactions on  Knowledge and Data Engineering, Volume:23, pp: 1432 – 1437, 2011.

[2] Kan Yang and Xiaohua Jia “An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing” ,IEEE Transactions on  Parallel and Distributed Systems, Volume:24,2012

[3] B. Wang, B. Li, and H. Li, “Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud,” IEEE Transactions on Cloud computing, pp. 43 - 56, 2014.

[4] C. Wang, S. S.-M. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy preserving public auditing for secure cloud storage.” , IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 2, pp: 362-375, FEBRUARY 2013

[4] Prasadreddy, P., T. Srinivasa and S. Phani,” A threat free architecture for privacy assurance in cloud computing.” Proceedings of the IEEE World Congress on Services, Jul. 4-9, IEEE Xplore Press, USA., pp: 564-568 , 2011.

[6] Ayad Barsoum and Anwar Hasan,“Enabling Dynamic Data and Indirect Mutual Trust for Cloud Computing Storage Systems”, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, pp: 2375 – 2385, 2013.

[7] Ranchal, R., B. Bhargava, L. Ben and L. Lilien, “Protection of identity information in cloud computing without trusted third party. “Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems, Oct. 31 Nov. -03, IEEE Xplore Press, Indian, pp: 368-372, 2010.

[8] C. Wang, Q. Wang, K. Ren, and W. Lou, “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,” in Proceedings of IEEE INFOCOM 2010, pp. 525–533 ,2010.

[9] B. Wang, B. Li, and H. Li, “Public Auditing for Shared Data with Efficient User Revoation in the Cloud,” in the Proceedings of IEEE INFOCOM 2013, pp. 2904–2912 , 2013.

[10]“Dynamic Audit Services for Outsourced Storages in Clouds”, IEEE Transactions on  Services Computing, Volume:6 , pp.227 – 238, 2013

[11] Sarfraz Nawaz Broh, “Design And Implementation Of A Privacy Preserved Off-Premises Cloud Storage”, Journal of Computer Science 10 (2): 210-223, 2014

[12] Boyang Wang, Baochun Li and Hui Li, “Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud”