International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.3, No.11, November 2014 DOI:10.15693/ijaist/2014.v3i11.74-81

A Survey on Various Cloud Storage Integrity Verifying Protocols

Rayees Ahmad Dar
M. Tech CSE
Pondicherry University

Kalyan Nandi
M. Tech CSE
Pondicherry University

Sebabrata Ghosh
M. Tech CSE
Pondicherry University

Vinod Kumar Yadav
M. Tech CSE
Pondicherry University

Abstract— **In cloud computing users outsource their data on remote cloud servers for storage and access it from these remote servers whenever required. However due to this outsourcing new security challenges need to be tackled. One of the main challenges is the ensuring the integrity of the data. There have been numerous attempts to provide and develop mechanisms and protocols which will ensure that the data on the remote servers preserve the integrity. In this paper we shall survey some of the prominent protocols developed in this regard against a set of parameters discussed.**

Keywords— **Integrity, storage auditing, dynamic auditing, batch auditing, provable data possession**

I. INTRODUCTION

Cloud computing is a type of Internet-based computing where computing services such as data, storage, applications, software and computing are delivered to local devices through Internet [8][12]. One of the important services provided by cloud computing is storage where large amounts of data are outsourced on cloud servers for storage, which is cost-effective and reliable. However there are inherent security threats that need to be addressed. In cloud the client is not in control of his data which brings the issue of confidentiality and integrity to the fore. Integrity implies that data should be honestly stored on cloud servers and any violations (e.g., data is lost, altered or compromised) are to be detected. Cloud servers are distrusted in terms of both security and reliability [6][11] because data could be lost from any infrastructure no matter what high degree of reliable measures cloud service providers (CSP) take. In some cases CSP may be dishonest as well. They could discard data that is rarely accessed motivated by saving storage space. Hence, owners must be convinced about the integrity of their data.

Extensive research has taken place in order to develop measures and protocols to ensure data integrity. In this paper we survey the various techniques that have been put forward to ensure integrity on cloud.

The rest of the paper is organised as follows: in Section II we shall give a literature survey, in Section III the parameters against which the various protocols are analysed will be discussed, in section IV the various protocols will be analysed, and finally conclusion will be given in Section V.

II. LITERATURE SURVEY

Data integrity is not a problem exclusive to cloud computing only, rather it is a common cryptographic problem. The traditional answer to it has been using the message authentication codes (MAC) or digital signature. However such strategies are not suitable for cloud environment since there is tremendous amount of data involved hence methods that require hashing for an entire file become prohibitive [11]. Further downloading a file for integrity verification is not possible as it will consume bandwidth and is computationally expensive as well.

However the integrity problem for cloud closely resembles with that of archival systems. The first attempt in this regard was taken by Ateniese et al. by proposing the Provable Data Possession (PDP) [2] protocol which for the first time provided a way to ensure integrity of data stored on remote servers without downloading it. A similar protocol called Proof of Retrievability (POR) [1] but with subtle differences was proposed by Juels et al. However these protocols were primarily designed for archival systems, they were not fully compliant with cloud environment. Nevertheless they provided a good base for future research. Almost all the subsequent protocols developed were a direct or indirect extension of PDP or POR.

One of the prominent requirements of cloud which makes it different from archival systems is the support for dynamic operations like append, modify, insert, delete etc. Simple PDP and POR were only supportive of static data hence subsequent research focussed on supporting dynamic data. Ateniese himself along with others proposed Scalable and Efficient PDP [6] to support dynamic operations. However, it involved pre-computing the set of challenges and only limited number of dynamic operations was supported. Erway et al. Proposed Dynamic PDP [3] to support dynamic operation fully, but at the cost of increased computational overhead.

The modern protocols are specially developed for cloud environment and take in consideration all the requirements of cloud. One of the notable features of these protocols is introduction of a third party auditor (TPA) for carrying out the auditing process (henceforth auditing protocol will mean a protocol intended for integrity verification of data on CSPs). [4][5][9][14][17] are worth mentioning in this regard and shall be surveyed in this paper.

III. PARAMETERS FOR ANALYSIS

There are certain inherent requirements that must be met by any integrity verifying protocol developed for the cloud computing. We present these parameters below:

A. Confidentiality

The integrity checking protocol should keep user's data confidential from anybody except for him including the cloud service provider (CSP). The requirement is more stringent when a third part auditor (TPA) is introduced for auditing services. In that scenario the data will flow between the user, TPA and CSP. Hence the protocol shall in no way reveal the data to either CSP or TPA.

B. Dynamic operations

The cloud storage service is fairly different from the archival systems. In the archival systems the data is static in that once data is stored it is not changed. However in the case of cloud, the data could be frequently modified a operations such as insert, delete, modify etc should be supported by the protocol.

C. Batch Auditing

The modern auditing protocols delegate the auditing services to the TPA, where there are multiple owners of data and there are multiple CSPs. At times an owner may use services of multiple CSPs. A TPA may receive auditing requests from multiple data owners. Hence the TPA must have the capability to combine all these requests together and only conduct the batch auditing for multiple owners simultaneously. Similarly in case a data owner has data on multiple CSPs, it should have the capability to combine responses from all the CSPs together and do batch verification [9].

D. Timely Detection

The auditing protocol should detect error or losses in the outsourced storage as well as anomalous behaviour of data operations in a timely manner. It should not be the case that auditing protocol detects any discrepancies at a time when the damage caused thereof is tremendous. The discrepancy detection should almost be instantaneous. This parameter is closely related to "*unforgeability*" which ensures that no dishonest CSP pass the audit test by any means without indeed keeping the users data intact.[16]

E. Light weight

The auditing protocol should not incur computational and storage overheads unbearably on either the owner or the TPA. In other words, the overheads should be as minimal as possible. Also the bandwidth consumption should be kept as low as possible [16].

IV. ANALYSIS

In this section we will analyse the various protocols that have been developed for integrity verification against the parameters discussed in the previous section.

A traditional approach to the data integrity problem is to employ MACs as mentioned in section II, where a small amount of MACs for the outsourced file is maintained by the user. At a later time when the user finds a need to query the server regarding the integrity of the file, the user can download the file and recalculate the MAC, thereby comparing the two for check if the file is intact or not.

Although this method ensures complete integrity, clearly it is not suitable for the cloud as it requires downloading of the file every time verification is to be carried out.

An improvement to the above problem could be to compute a number of MACs on different keys and then query the server by releasing one key per query. The computation can be delegated to a trusted third party (TTP). But the evident drawback to this improvement is that once the keys are over, new MACs need to be calculated once again. Further dynamic operations are not supported as any change to the file would render all of the previously calculated MACs invalid [5].

Researchers have proposed the probabilistic solutions, rather than deterministic ones, to the integrity problem taking into account the problem and constraints discussed above.

The general idea of these schemes is that some metadata is initially generated on the file blocks and stored locally (or on a TTP) as well as with the file at the server. Later on the client can query the server for certain blocks of the file randomly. Using the queried blocks and their tags the server computes a proof and sends it back to the client and then checked by him to verify the integrity of the file.

So we can divide the operation of these protocols into two stages as below:

A. Initialization:

In this stage the cryptographic keys and the metadata are computed. Two algorithms are involved in this stage.

1) `KeyGen`*:* This algorithm generates the keys for the protocol. The generated keys depend upon the scheme to be used, i.e., symmetric key or asymmetric key cryptography. Also the number of keys to be generated too depends upon the protocol itself.

2) `TagGen`*:* This algorithm is responsible for generating the tags (metadata) on the individual blocks (or a selected set of blocks) using the keys generated in `KeyGen` algorithm. Actually the file is divided into individual blocks and tags are generated on them. Finally the file along with the tags is sent to the server for storage.

B. Verification:

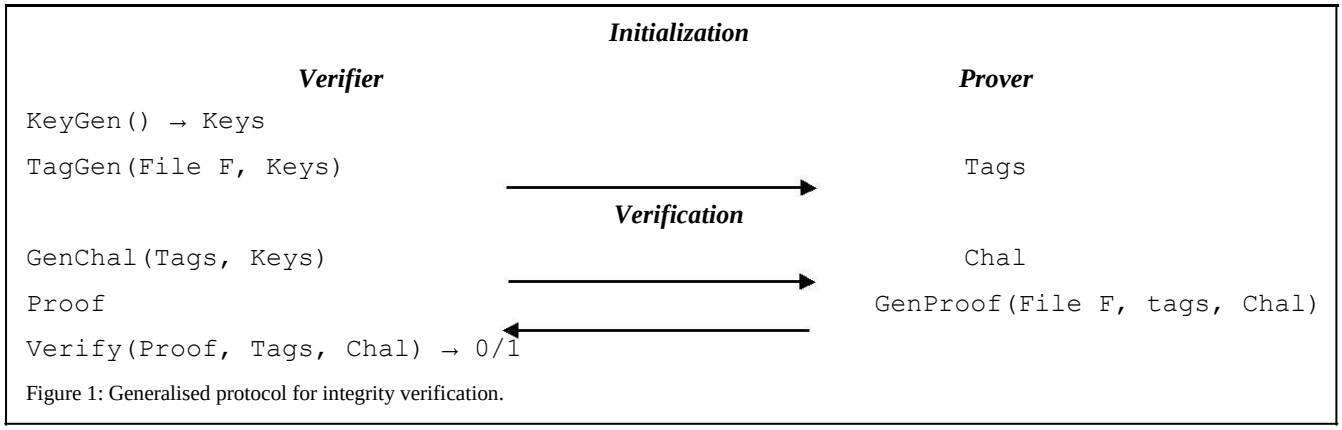

Figure 1: Generalised protocol for integrity verification.

Whenever the verifier finds a need to verify the integrity of the file, he issues a query to the prover, the prover computes the response and sends it back to the verifier who verifies the response. Hence, three algorithms are involved in this phase. Some protocols like the CPDP[17] involve more algorithms but they can be adopted to this generalised algorithm. Further the individual protocols have different nomenclature for these algorithms, which will be pointed out in their respective discussions.

1) `GenChal`*:* The algorithm is responsible for generating a challenge/ query to the prover to prove that a certain set of blocks are intact. This algorithm takes as input the metadata/tags and keys generated in the initialization phase.
2) `GenProof`*:* The prover on receiving the challenge from the verifier computes the response/proof from the blocks queried and the tags associated with them and sends it back to the verifier. Keys are used to generate responses in this stage also.
3) `Verify`*:* The verifier upon receiving the proof from the prover verifies it.

In the above discussion we have presented a general prover-verifier model where "prover" is the server however the "verifier" could be the data owner himself or a third part auditor (TPA). Hence two models are considered for the protocol presented in figure 1. The two models are depicted in figure 2 and figure 3. We shall first discuss about those protocols which involve only the data owner and the server. Later on we shall survey the protocols which involve a TPA to carryout the auditing process.

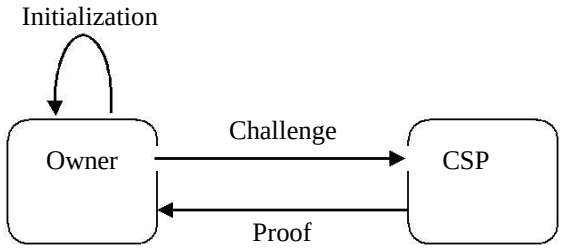

Figure 2 System Model for non-TPA based Auditing Protocols..

A. Non – TPA Based Protocols:

The system model for these types of protocols is given in figure 2. Below we survey some of the popular auditing protocols under this model. We shall conclude this discussion with the summary of the protocols under this category.

1) *Provable Data Possession (PDP) [2]:* This is one of the first protocols (along with POR discussed next) to ensure data integrity over remote server without the need to download data. It follows the algorithm given in figure 1 with the following details:
   a) `KeyGen` uses public key cryptography.
   b) It uses Homomorphic Verifiable Tags (HVT) which are unforgeable tags generated upon file blocks to act as verification metadata for the file blocks. The HVT allows blockless verification, i.e., the prover can construct a proof that allows the verifier to verify if the prover possesses certain file blocks even when the verifier does not have access to actual blocks. The HVTs have the property that given two values T_{m1} and T_{m2} anyone can combine them into a value T_{m1+m2} corresponding to *m1+m2*.
   c) The challenges generated by `GenChal` of figure 1 are random ensured by a pseudorandom function (PRF).

   The scheme was however developed for static data and hence provides no support for dynamic operations. Further the protocol was built for largely public data like libraries, i.e., anyone can access the data on the server and hence no measure for confidentiality was incorporated.

2) *Proof of Retrievability (POR [1][7][10]):* This protocol was developed along with the PDP protocol. However, this protocol, in addition to integrity verification has error-correcting capabilities and ensures that file is retrieved from the server. The protocol with respect to the general protocol given in figure 1 has the following details:

a) `KeyGen` uses symmetric key cryptography.

b) The `TagGen` algorithm has four parts:

b.1) File F is divided into *K*-block chunks and an error correcting code is applied to each chunk.

b.2) File F' is encrypted using the symmetric cipher key generated in `KeyGen`.

b.3) s sentinels are created using a suitable one-way function and then these sentinels are appended to the file F'. The sentinels are just like normal file blocks.

b.4) The blocks of the encrypted file along with sentinel blocks are permuted using a pseudorandom permutation (PRP). This makes the sentinels indistinguishable from normal blocks.

c) During the verification phase, the verifier reveals the location of a set of sentinels using the PRP and requests the prover to return the sentinels. The verifier later verifies the proof sent by prover.

The idea behind the protocol is that if the server has modified or deleted a substantial portion of the outsourced file, then with high probability it would also have suppressed a number of sentinels, resulting in server's inability to respond correctly.

A direct disadvantage of the protocol is that it runs only a bounded number of times as the number of sentinels are fixed a priori and at each verification a subset of sentinel is revealed, hence rendering them unusable. However later schemes attempted to overcome this limitation [7]. Like PDP, this protocol also supports only static data and has no support for data dynamics.

3) *Scalable and Efficient PDP (SPDP) [6]:* As mentioned above both PDP and POR were suitable for static data only. SPDP attempted to add support for dynamic operations as well as reducing some of the computational overhead over the client. This protocol is basically a modification of the original PDP with two basic differences that (1) it is based on entirely symmetric key cryptography, thereby lowering the computational complexity on the client side, and (2) allows outsourcing of dynamic data.

The main approach to support dynamic operations is that it requires all challenges to be pre-computed at setup phase. Hence the number of challenges are limited which could result in the server deceiving the owner by using previous metadata or responses due to lack of randomness in the challenges [17]. Further the number of updates is limited, any update requires recomputing the whole metadata and the dynamic operation support is also minimal as there is no support for block insertions anywhere [9].

4) *Dynamic PDP (DPDP) [3]:* This is also the modification of the original PDP to fully allow dynamic operations (insert, modify and delete). The DPDP introduces three new operations known as *PrepareUpdate*, *PerformUpdate* and *VerifyUpdate*. *PrepareUpdate* is run by the owner to generate a request for update which includes the updates to be performed. An example of this request could be delete block *i*, update block *i* etc. *PerformUpdate* is run by the server to carry out the actual update. After running the *PerformUpdate* the server returns the update proof and the owner runs the *VerifyUpdate* to verify if the update has been performed correctly.

It introduces rank-based authenticated skip-lists (ASL) for blockless verification and data dynamics. The nodes of the rank-based ASL contain tags. In the `GenChal` algorithm the prover is asked to provide the random tag T_i and its path from root of ASL. The verifier then verifies to check if the block associated with the tag is correctly stored or not based on path returned by `GenProof` algorithm.

It must be noted that the ASL stores tags which are authenticated by the skip-list itself while as the tags authenticate the blocks. The `TagGen` algorithm is modified here to support data dynamics. The index information is removed from the tag computation (as in PDP model) and ASL is used instead to authenticate tag information of challenged or updated blocks. The `TagGen` also returns the root of the ASL to the verifier.

However, this scheme may cause heavy computation burden to the server [9]. Also data blocks may be leaked by the response of a challenge [17]. Further there is no scope for batch auditing for multi-cloud multi-owner cloud environment.

5) *Summary of non-TPA Based Protocols:* PDP (including POR – based protocols) is a class of problems that provides efficient and practical approaches to verify the integrity of a file stored on remote servers, without downloading the file. These protocols evolved from supporting only static data to dynamic operations. However since the verification is only between client and the server, there is some burden upon the client in terms of computation and storage. This is particularly worrying in cloud model as most of the clients are thin clients only with limited storage and computational capabilities. Further as the cloud has evolved into a multi-owner, multi-cloud model, the non-TPA based approaches fail to adapt to it, particularly the batch auditing. Hence, a need arises there to move from non-TPA to TPA based solutions.

A. TPA – Based Protocols:

As mentioned above the cloud model today is a multi-cloud multi-owner environment, batch auditing becomes a compelling factor to be incorporated into the integrity verification protocols for the cloud. Further considering the nature of the clients, modern auditing protocols are focussing much on third part auditing (TPA). This is intended to relieve the client of most of the computational and storage needs and such complexities will be delegated to the TPA.

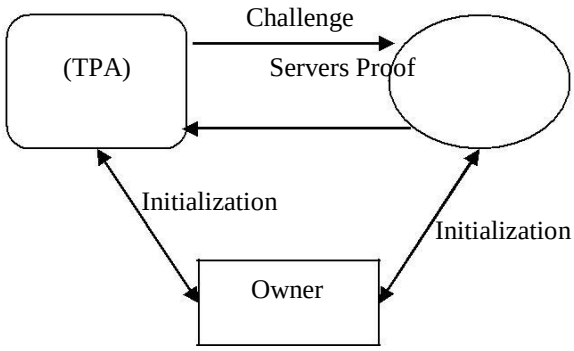

Figure 3 System Model for Protocols with TPA

However, the TPA cannot be directly added to the previous solutions. The reason is that in PDP – based solutions the prover responds with a proof using HLA (Homomorphic Linear Authenticator), which can potentially reveal the user data to the TPA and violate the privacy preserving guarantee. HLA is nothing but a linear combination of blocks $= \sum$. Specifically by challenging the same set of blocks $m_1, m_2, \ldots, m_c$ using c different sets of random coefficients $\{v_i\}$, TPA can accumulate c different linear combinations $\mu_1, \mu_2, \ldots, \mu_c$. With $\{\mu_i\}$ and $\{v_i\}$, TPA can derive the data $m_1, m_2, \ldots, m_c$ by simply solving a set of linear equations [4].

Therefore, solution have been put forward in this regard which will be discussed below. The system model is given in figure 3.

1) *Privacy Preserving Public Auditing [4][5][14]:* In [14], the authors proposed the use of HLA to support public auditability along with MHTs (Merkel Hash Trees) where leaf nodes of the tree are ordered set of hashes of "file tags" $H(m_i)$ for data dynamics. However, as pointed out for the DPDP scheme, this solution also suffers from the privacy problem. The authors extended their scheme to be privacy preserving in [4]. In their privacy preserving scheme they coupled the homomorphic authenticators with masking. The details of their solution with respect to generalised protocol of figure 1 are:

a) `KeyGen` generates the public and secret parameters. Specifically the user chooses a random signing key pair (*ssk, spk)* for signing the root of the MHT.

b) In the `TagGen` (here called `SigGen`) authenticators for each block are computed and the set of authenticators are denoted by Φ. The `SigGen` also computes a file tag t which is a combination of the filename and its signature. The pair (Φ, t) is sent to the server. The root of the MHT is sent to the TPA.

c) In the verification phase (here called as Audit), the TPA first retrieves the file tag *t*, and verifies it using *spk* thereby retrieves the filename if the verification is successful.

d) Now a random challenge is generated specifying the portion of blocks required to be audited.

e) The server in `GenProof` generates the proof. However, here the masking is used to protect the privacy of data against TPA. Also the AAI (Auxiliary Authentication Information) of challenged blocks is sent.

f) In `Verify` phase the TPA first uses the root of MHT and AAI to authenticate the tags and later tags are then used to verify the response.

In [9] the authors claim that this scheme may leak the data content to the auditor since it requires the prover to send linear combination of data clocks to the auditor. In [4] the authors extended their dynamic auditing scheme to be privacy preserving and supportive of batch auditing for multi-owner environment. However it incurs a heavy storage overhead on the server since a large number of data tags are involved.

2) *Cooperative PDP (CPDP) [17]:* To address the problem of high storage overhead in [4], Zhu et al. introduced a fragment structure in [16] where a file is split into *n* blocks and each block is further split into *s* sectors. Here a tag is still generated for each block; however we can reduce the number of tags generated by increasing the number of sectors in each block.

For dynamic operations, they proposed the use of IHT (Index Hash Table) to record changes of file blocks as well as to generate the hash value of each block during verification phase. The IHT consists of records which contain serial number, block number, version number and a random integer. The structure of the IHT is similar to the block allocation table in file systems.

Later on the authors in [17] extended the same concept to have incorporated the support for batch auditing.

Table 1
Comparison of Various Integrity Checking Protocols

Property / Paper	Confidentiality	Dynamic operations	Batch auditing		Computational Complexity		Communication complexity	Unforgeability	Prob. Of detection
			Multi-owner	Multi-cloud	Server	Verifier			
PDP[2]	Yes	No	No	No	$O(t)$	$O(t)$	$O(1)$	Yes	$1 - (1 - \rho)^t$
POR[1]	Yes	No	No	No	$O(t + s)$	$O(t + s)$	$O(t + s)$	Yes	$1 - (1 - \rho)^{ts}$
SPDP[6]	Yes	Partial	No	No	$O(t)$	$O(t)$	$O(1)$	Vulnerable	$1 - (1 - \rho)^t$
DPDP[3]	Vulnerable	Yes	No	No	$O(t \log n)$	$O(t \log n$	$O(t \log n)$	Yes	$1 - (1 - \rho)^t$
Audit[4][5]	Vulnerable	Yes	Yes	Yes	$O(t \log n$	$O(t \log n$	$O(t \log n)$	Yes	$1 - (1 - \rho)^t$
CPDP[15]	Yes	Yes	No	Yes	$O(ts)$	$O(t + s)$	$O(t + s)$	Vulnerable	$1 - (1 - \rho)^{ts}$
Dynamic Auditing[8]	Yes	Yes	Yes	Yes	$O(ts)$	$O(t)$	$O(t)$	Yes	$1 - (1 - \rho)^{ts}$

n is the total number of data blocks of a file; t is the number of challenged data blocks in an auditing query; s is the number of sectors in each data block; ρ is the probability of block/sector corruption

One of the striking features of the protocol is the hierarchy structure that provided a virtualization approach to conceal the storage details of multiple CSPs. This hash-index hierarchy (figure 4) maps efficiently with the multi-cloud model. It provides three layers of abstraction for storage as:

a) *Express Layer* provides abstract representation of the stored resources.

b) *Service Layer* offers and manages cloud services. This layer exposes the various CSPs.

c) *Storage Layer* realises data storage on many physical devices. It exposes the actual storage infrastructure.

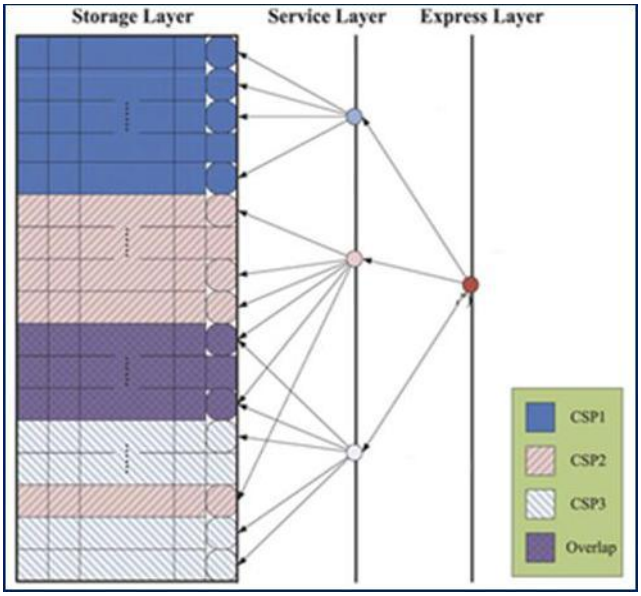

Figure 4 Index hash hierarchy of CPDP model

To support the multi-cloud model they extended the concept of HVT to responses, where given two responses θ_i and θ_j for two challenges Q_i and Q_j from two CSPs, there exists an efficient algorithm to combine them into a response θ corresponding to the sum of challenges $\cup$. This reduces the communication complexity as well as conceals the location of outsourced data in multi-cloud environment. With respect to generalised protocol of figure 1, CPDP has following details:

a) `KeyGen`: is used again to generate public – private keys.

b) `TagGen`: Here the file is split into × sectors and the tags are generated for each block taking into account the hierarchy structure provided. It constructs the hash-table as well. Here the TTP stores the index-table and the tag information while as CSP is uploaded with the file along with tags. The data owner saves the secrets used to generate the tags.

c) Verification is a five stage algorithm where `GenChal` (of figure 1) is split into three algorithms *commitment, challenge1* and *challenge2* as below:

c.1) The organiser initiates the protocol and sends a commitment to the verifier.

c.2) The verifier returns a challenge to the organiser.

c.3) The organiser relays the challenge to each CSP according to exact position of each data block.

d) The `GenProof` is also split into two algorithms, *Response1* and *Response2* as below:

d.1) Each CSP generates response to the challenge received and returns the same to the organiser.

d.2) The organiser aggregates all the challenges using HVR and forwards to the owner.

e) In the Verify algorithm the owner verifies the aggregated response.

Wang et al. in [15] argue about the knowledge soundness of the system and claim that any attacker can get the pay without storing the client's data. Hence this scheme is vulnerable to forgeability attack.

Yang et al. [9] argue that it is impossible for this scheme to support batch auditing for multiple owners because parameters for generating tags for each owner are different and hence data tags cannot be combined from multiple owners to conduct batch

auditing. Also the introduction of the organiser is not practical in cloud storage systems.

3) *Dynamic Auditing [9]:* This protocol adopts the fragment structure of [16][17] and provides support for batch auditing to multi-owner environment as well. However the hierarchy structure is not used here, since it conceals the actual CSPs from the users. For data dynamics IHT is again used. With respect to protocol given in figure 1, the details of this scheme are:
   a) `KeyGen`: It generates the secret tag and hash keys and a public tag key.
   b) `TagGen`; It generates data tags as before and also generates IHT.
   c) `GenChal`: The auditor runs this algorithm to generate a random challenge for the CSP. For batch auditing, it generates a random challenge for each CSP as an aggregation of challenges corresponding to each owner. The challenges are then delegated to each CSP.
   d) `GenProof`: Each CSP generates the proof for each challenge as a combination of data proof and tag proof.
   e) `Verify`: Here the TPA verifies the responses received from CSPs.

   In [13] Ni et al. have shown that that the protocol is insecure when an active adversary is involved in the cloud environment that can fool the TPA by modifying the proof that data is correctly stored while it wouldn't be. They also proposed a solution to fix the problem by employing digital signature to prevent proof from being modified.

   This scheme is the best scheme which has all the required qualities (after fixing the problem pointed out in [13]) and as of now is the promising protocol

4) *Summary of TPA – based Protocols:* These solutions have the capability of relieving the owner of storage and computational complexities. Also TPA – based approaches are more suitable for today's cloud environment. The protocols have adopted the dynamic capabilities from existing non-TPA – based solutions and also added the batch auditing for complete auditing solution.

V. CONCLUSION

Integrity is the main issue with storage as are service in cloud environments. The PDP scheme laid the foundation for developing the auditing protocols which can verify the integrity of a file without downloading the actual file. This basic scheme has been since modifies to correctly model the cloud environment. Support for data dynamics and multi-owner, multi-cloud, i.e., batch auditing has been added by various researchers starting from Wang et al. in [4][5][14][16] through [17] to [9]. [9] as of now singles itself out to be the most promising protocol to enforce integrity in the cloud. However there is scope for further research in minimizing the bandwidth and computational and storage complexity of clients.

Table 1 provides the comparison of various protocols surveyed in this paper against the parameters discussed in section III.

REFERENCES

[1] Ari Juels and Burton S. Kaliski, Jr.. 2007. Pors: proofs of retrievability for large files. InProceedings of the 14th ACM conference on Computer and communications security (CCS '07). ACM, New York, NY, USA, 584-597.

[2] Ateniese, Giuseppe; Burns, Randal; Curtmola, Reza; Herring, Joseph; Kissner, Lea; Peterson, Zachary; and Song, Dawn, "ProvableData Possession at Untrusted Stores" (2007). Department of Electrical and Computer Engineering.Paper 37.

[3] Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia. 2009. Dynamic provable data possession. In Proceedings of the 16th ACM conference on Computer and communications security (CCS '09). ACM, New York, NY, USA, 213-222.

[4] Cong Wang; Chow, S.S.M.; Qian Wang; Kui Ren; Wenjing Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," Computers, IEEE Transactions on , vol.62, no.2, pp.362,375, Feb. 2013,

[5] Cong Wang; Kui Ren; Wenjing Lou; Jin Li, "Toward publicly auditable secure cloud data storage services," Network, IEEE , vol.24, no.4, pp.19,24, July-August 2010

[6] Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik. 2008. Scalable and efficient provable data possession. In Proceedings of the 4th international conference on Security and privacy in communication netowrks (SecureComm '08). ACM, New York, NY, USA, , Article 9 , 10 pages.

[7] Jiawei Yuan and Shucheng Yu. 2013. Proofs of retrievability with public verifiability and constant communication cost in cloud. In Proceedings of the 2013 international workshop on Security in cloud computing (Cloud Computing '13). ACM, New York, NY, USA, 19-26.

[8] Stanoevska-Slabeva, T. Wozniak, and S. Ristol "Grid and cloud computing- a business perspective on technology and applications," Springer-Verlag, Berlin, Heidelberg, 2009.

[9] Kan Yang; Xiaohua Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing," Parallel and Distributed Systems, IEEE Transactions on , vol.24, no.9, pp.1717,1726, Sept. 2013

[10] Kevin D. Bowers, Ari Juels, and Alina Oprea. 2009. Proofs of retrievability: theory and implementation. In Proceedings of the 2009 ACM workshop on Cloud computing security (CCSW '09). ACM, New York, NY, USA, 43-54.

[11] Minqi Zhou; Rong Zhang; Wei Xie; Weining Qian; Aoying Zhou, "Security and Privacy in Cloud Computing: A Survey," Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on , vol., no., pp.105,112, 1-3 Nov. 2010

[12] National Institute of Standards and Technology, "The NIST definitionof cloud computing," Information Technology Laboratory, 2009.

[13] Ni, J.; Yu, Y.; Mu, Y.; Xia, Q., "On the Security of an Efficient Dynamic Auditing Protocol in Cloud Storage," Parallel and Distributed Systems, IEEE Transactions on , vol.PP, no.99, pp.1,1, 0

[14] Qian Wang; Cong Wang; Kui Ren; Wenjing Lou; Jin Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," Parallel and Distributed , IEEE Transactions on , vol.22, no.5, pp.847,859, May 2011

[15] Wang, Huaqun; Zhang, Yuqing, "On the Knowledge Soundness of a Cooperative Provable Data Possession Scheme in Multicloud Storage," Parallel and Distributed Systems, IEEE Transactions on , vol.25, no.1, pp.264,267, Jan. 2014

[16] Yan Zhu; Gail-Joon Ahn; Hongxin Hu; Yau, S.S.; An, H.G.; Chang-Jun Hu, "Dynamic Audit Services for Outsourced Storages in Clouds," Services Computing, IEEE Transactions on , vol.6, no.2, pp.227,238, April-June 2013.

[17] Yan Zhu; Hongxin Hu; Gail-Joon Ahn; Mengyang Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage," Parallel and Distributed Systems, IEEE Transactions on , vol.23, no.12, pp.2231,2244, Dec. 2012

Authors Profile

Rayees A. Dar is currently pursuing his **M.Tech.** in Computer Science and Engineering from Pondicherry University, Pondicherry, India. He was formerly Assistant Professor in the department of Computer Science and Engineering at IUST, J&K, India. He has received his **B.Tech (CSE)** from Islamic University of Science and Technology, J&K, India. His research interests include Natural Language Processing, Text Analysis, Plagiarism Detection, Assistive Computing and Information Retrieval.

Kalyan Nandi received the B.Sc. degree in Computer Applications (Major) from the Ramakrishna Mission Vidyamandira College, Belur, Calcutta University, West Bengal, India, in 2011 & M.Sc. degree in Computer Science from Pondicherry University, Pondicherry, India. He is currently doing **M.Tech**. in Computer Science and engineering from Pondicherry University, Pondicherry, India. His research interests include Bio Inspired Algorithm, Machine learning, Distributed System & Cloud Computing.

Sebabrata Ghosh received the B.Sc. degree in Computer Applications (Major) from the Ramakrishna Mission Vidyamandira College, Belur, Calcutta University, West Bengal, India, in 2011 & M.Sc. degree in Computer Science from Pondicherry University, Pondicherry, India. He is currently doing **M.Tech**. in Computer Science and Engineering from Pondicherry University, Pondicherry, India. His research interests include Bio Inspired Algorithms, Machine learning, Artificial Intelligence and Web Technology.

Vinod Kumar received the B.Sc. degree from Erwing Christain College, Allahabad, University of Allahabad, India, in 2008 & M.C.A. degree from IGNOU, India. Pondicherry University, Pondicherry, India. He is currently doing **M.Tech**. in Computer Science and Engineering from Pondicherry University, Pondicherry, India. His research interests include Artificial Intelligence, Computer Vision, Information Fusion and Distributed Computing.