ISSN:0975-9646

Proving Possession and Retrievability within a Cloud Environment: A Comparative Survey

Kochumol Abraham, Win Mathew John

P G Department of Computer Applications,
Marian College, Kuttikkanam P O, Idukki Dist, Kerala, India

***Abstract*— Outsourcing data to a remote Cloud Service Provider (CSP) is a growing trend for numerous organizations alleviating the burden of local data storage and maintenance. While Cloud computing makes these advantages more appealing than ever, it also brings new challenging security threats towards user's outsourced data. It's of crucial importance to customers to have strong evidence that they actually get the service they pay for. Moreover, they need to verify that all their data copies are not being tampered with or partially deleted over time. As a result, data owners need to be convinced that their data are correctly stored in the Cloud. So, one of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. In order to solve the problem of data integrity checking, many schemes are proposed under different systems and security models. In this paper we surveyed two core integrity proving schemes in detail along with different methods used for data integrity in both the schemes.**

***Keywords*— Cloud Service Provider, Provable Data Possession, Proof of Retrievability.**

I. INTRODUCTION

Data outsourcing brings with it many advantages. But associated with it are the risks involved. Though client cannot physically access the data from the cloud server directly, without client's are either not used by client from a long time. Hence, there is a requirement of checking the data periodically for correction purpose, known as data integrity. Here we provide a survey on the different techniques of data integrity. The basic schemes for data integrity in cloud are Provable Data Possession (PDP) knowledge, cloud provider can modify or delete data which and Proof of Retrievability (PoR). These two schemes are the most active area of research in the cloud data integrity field.

II. PDP AND POR

To restore security assurances eroded by cloud environments, researchers have proposed two basic approaches to client verification of file availability and integrity. The cryptographic community has proposed tools called proofs of retrievability (PORs) and proofs of data possession (PDPs).PDP scheme checks that a remote cloud server retains a file, which consists of a collection of n blocks. The data owner processes the data file to generate some metadata to store it locally. The file is then sent to the server, and the owner delete the local copy of the file. The owner verifies the possession of file in a challenge response protocol. A POR is a challenge response protocol that enables a prover (cloud-storage provider) to demonstrate to a verifier (client) that a file *F* is retrievable, i.e., recoverable without any loss or corruption. The benefit of a POR over simple transmission of *F* is efficiency. The response can be highly compact (tens of bytes), and the verifier can complete the proof using a small fraction of *F*.As a standalone tool for testing file retrievability against a single server, though, a POR is of limited value. Detecting that a file is corrupted is not helpful if the file is irretrievable and the client has no recourse. Thus PORs are mainly useful in environments where *F* is distributed across multiple systems, such as independent storage services. In such environments, *F* is stored in redundant form across multiple servers.

III. RELATED WORK

A. Static Provable Data Possession (PDP)

The fundamental goal of the PDP scheme is to allow a verifier to efficiently, periodically, and securely validate that a remote server which is supposed to store the owner's potentially very large amount of data is not cheating the verifier. The problem of data integrity over remote servers has been addressed for many years and there is a simple solution to tackle this problem as follows.

1) Basic PDP Scheme based on MAC : The data owner computes a Message Authentication Code (MAC) of the whole file before outsourcing to a remote server. Keeps only the computed MAC on his local storage, sends the file to the remote server, and deletes the local copy of the file. Later, whenever a verifier needs to check the data integrity, he sends a request to retrieve the file from the archive service provider, re-computes the MAC of the whole file, and compares the re-computed MAC with the previously stored value**.**

2) PDP Schemes based on functions f and $H^{|}$**.** $H^{|}$ is a one-way function, f is another function such that f(C, $H^{|}$(File)) = H (C||File), where H - Secure hash function and C - Random challenge number sent from the verifier to the remote server. [2]

The protocol is as follows:

Data owner computes $H^{|}$ (File) and store it on the local storage.[2]

To audit the file, the verifier generates a random challenge C,

Computes V = f(C, $H^{|}$ (File)), and sends C to the remote server.

The server computes R = H(C||File) and sends the response R to the verifier.

To validate the file integrity, the verifier checks V ?= R.

3) RSA Based PDP Schemes:
RSA-based Homomorphic hash function[3]

A function H is Homomorphic if, given two operation + and x, we have H ($d+d^{|}$) = H(d) x H($d^{|}$). The response R = H(d) is a homomorphic function in the data file d; H(d + $d^{|}$) = $r^{d+ d|}$ r^{d} $r^{d|}$= H(d)H($d^{|}$) mod N.[3] To find a collision

for this hash function, one has to find two messages d, d′ such that $r^d \equiv r^{d'}$, i.e., $r^{d-d'} \equiv 1$ *mod N*. Thus, d − d′ must be multiple of ϕ(N). Finding such two messages d, d′ is believed to be difficult since the factorization of N is unknown.

Limitations

The archive service provider has to exponentiate the entire data file plus the storage overhead on the verifier side. Solution is to use an RSA-based hash function on the blocks.

RSA-based hash function on the blocks

Fragment the file into blocks

Fingerprint each block, and then use an RSA-based hash function on the blocks.

Thus, the file F is divided into a set of m blocks: F = {b1, b2, . . . , bm}, where m fingerprints {Mi}1¡Âi¡Âm are generated for the file and stored on the verifier local storage. Their proposal does not require the exponentiation of the entire file.

4) Data Storage Commitment Schemes:

A storage-enforcing commitment scheme (SEC) is a three-party protocol executed between a message source S, a prover P, and a verifier V. [4]

The message source communicates the message M to the prover and the commitment C to the verifier.

The verifier V may verify whether the prover is storing the secret by invoking a probabilistic interactive algorithm. This algorithm may be executed an unlimited number of times.

Once the message is revealed, the verifier may check the commitment by running the algorithm Verify.

This scheme has three properties called binding, concealing, and storage-enforcing.

5) Privacy-Preserving PDP Schemes:

The data owner first encrypts the file,

Sends both the encrypted file along with the encryption key to the remote server.

Moreover, the data owner sends the encrypted file along with a key-commitment that fixes a value for the key without revealing the key to the TPA.

The primary purpose of this scheme is to ensure that the remote server correctly possesses the client's data along with the encryption key, and to prevent any information leakage to the TPA which is responsible for the auditing task[5]. Thus, clients especially with constrained computing resources and capabilities can resort to external audit party to check the integrity of outsourced data, and this third party auditing process should bring in no new vulnerabilities towards the privacy of client's data. In addition to the **auditing task** of the TPA, it has another primary task which is **extraction of digital contents**. [6]

6) PDP in Database Context based on signature aggregation :

Each database record is signed before outsourcing the database to a remote service provider.[7]

2 aggregation mechanisms:

Scheme based on RSA [8] and

Scheme based on BLS signature [9]

Scheme based on the RSA signature

Each record in the database is signed as: σi = h(bi)d mod N where h is a one-way hash function, bi is the data record, d is the RSA private key, and N is the RSA modulus.

A user issues a query to be executed over the outsourced database, the server processes the query and computes an aggregated signature σ = Σti=1 σi mod N, where t is the number of records in the query result. The server sends the query result along with the aggregated signature to the user. To verify the integrity of the received records, the user checks σe = πti=1 σi mod N, where e is the RSA public key.

Scheme based on the BLS signature is similar to the first scheme but the record signature σi = h(mi)x, where x ∈R Zp is a secret key.

7) PDP Schemes Based on Homomorphic Variable Tags(HVTs)/Homomorphic Linear Authenticators(HLAs) :

HVTs/HLAs are unforgeable verification metadata constructed from the file blocks in such a way that the verifier can be convinced that a linear combination of the file blocks is accurately computed by verifying only the aggregated tag/authenticator.

Public verifiability and private verifiability.

In public verifiability anyone not necessarily the data owner who knows the owner's public key can challenge the remote server and verify that the server is still possessing the owner's files. On the other side, private verifiability allows only the original owner (or a verifier with whom the original owner shares a secret key) to perform the auditing task.

Two main PDP schemes

Sampling PDP (S-PDP) and Efficient PDP (E-PDP) schemes. [10]

Based on KEA1 assumption (Knowledge of Exponent Assumption). It focuses on the problem of auditing if an untrusted server stores a client's data.

B. Dynamic Provable Data Possession (DPDP)

The PDP schemes discussed above focus on static or warehoused data which is essential in numerous different applications such as libraries, archives, and astronomical /medical /scientific /legal repositories. On the other side, Dynamic Provable Data Possession (DPDP) schemes investigate the dynamic file operations such as update, delete, append, and insert operations. There are some DPDP constructions in the literature satisfying different system requirements.

1) Scalable DPDP:

This scheme is based entirely on symmetric-key cryptography. [11]

1. Before outsourcing, data 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.
2. Subsequently, when data owner wants to obtain a proof of data possession, it challenges server with a set of random-looking block indices.
3. In turn, server must compute a short integrity check over the specified blocks (corresponding to the indices) and return it to data owner.

For the proof to hold, the returned integrity check must match the corresponding value pre-computed by data owner. However, in this scheme data owner has the choice of either keeping the pre-computed tokens locally or outsourcing them – in encrypted form – to server. Notably, in the latter case, data owner's storage overhead is constant regardless of the size of the outsourced data. This scheme is also very efficient in terms of computation and bandwidth.

2) DPDP-I:

Given a file F consisting of n blocks, we define an update as either insertion of a new block (anywhere in the file, not only append), or modification of an existing block, or deletion of any block. Therefore the update operation describes the most general form of modifications a client may wish to perform on a file. DPDP solution is based on a new variant of authenticated dictionaries, where rank information is used to organize dictionary entries[12]. Thus its able to support efficient authenticated operations on files at the block level, such as authenticated insert and delete. The security of the constructions using standard assumptions has been proved.

3) DPDP II:

The only difference between the two schemes is the authenticated structure used for protecting the integrity of the tags. It has a higher probability of detection and maintains logarithmic communication complexity but has increased update time[13]. A dynamic authenticated data structure called RSA tree is presented here that achieves constant expected query time (i.e., time to construct the proof), constant proof size, and $O(n^{\epsilon} \log n)$ expected amortized update time, for a given $0 < \epsilon < 1$. We can add rank information to the RSA tree by explicitly storing ranks at the internal nodes. Using this data structure allows the server to answer O (log n) challenges with O(log n) communication cost since the proof for a block tag has O (1) size.

4) Schemes in Hybrid clouds : Collaborative PDP

Homomorphic verifiable response is the key technique of collaborative PDP because it not only reduces the communication bandwidth, but also conceals the location of outsourced data in hybrid clouds. The collaborate integrity verification for distrusted outsourced data, in essence, is a multi-prover interactive proof system (IPS), so that the corresponding construction should satisfy the security requirements of IPS. Moreover, in order to ensure the security of verified data, this kind of construction is also a Multi-Prover Zero-knowledge Proof (MPZKP) system which can be considered as an extension of the notion of an interactive proof system.

Given an assertion L, such a system satisfies three following properties:

Completeness: whenever $x \in L$, there exists a strategy for provers that convinces the verifier

Soundness: whenever $x 6\in L$, whatever strategy the provers employ, they will not convince the verifier that $x \in L$;

Zero-knowledge: no cheating verifier can learn anything other than the veracity of the statement.

Scalia

A cloud storage brokerage solution that continuously adapts the placement of data based on its access pattern and subject to optimization objectives, such as storage costs. Scalia efficiently considers repositioning of only selected objects that may significantly lower the storage cost.Scalia can run directly at the customer premises as an integrated hardware and software solution (i.e., an appliance) or can be deployed as a hosted service across several data centers, putting the emphasis on providing a scalable and highly available architecture with no single point of failure, able to guarantee higher availability than the storage providers.

C. Multi-Copy PDP Schemes (MC-PDP Schemes*)*

Suppose that a CSP offers to store n copies of an owner's file on n different servers to prevent simultaneous failure of all copies. Thus, the data owner needs a strong evidence to ensure that the CSP is actually storing no less than n copies, all these copies are complete and correct, and the owner is not paying for a service that he does not get. A solution to this problem is to use any of the previous PDP schemes to separately challenge and verify the integrity of each copy on each server. This is certainly not a workable solution; cloud servers can conspire to convince the data owner that they store n copies of the file while indeed they only store one copy.

Whenever a request for a PDP scheme execution is made to one of the n severs, it is forwarded to the server which is actually storing the single copy. The CSP can use another trick to prove data availability by generating the file copies upon a verifier's challenge; however, there is no evidence that the actual copies are stored all the time. The main core of this cheating is that the n copies are identical making it trivial for the servers to deceive the owner. Therefore, one step towards the solution is to leave the control of the file copying operation in the owner's hand to create unique distinguishable/differentiable copies.

1) Basic Multi-Copy Provable Data Possession (BMC-PDP) scheme :

The data owner creates n distinct copies by encrypting the file under n different keys keeping these keys secret from the CSP. Hence, the cloud servers could not conspire by using one copy to answer the challenges for another. This natural solution enables the verifier to separately challenge each copy on each server using any of the PDP schemes, and to ensure that the CSP is possessing not less than n copies.

2) Multiple-Replica Provable Data Possession (MR-PDP) scheme :

Creating distinct replicas/copies of the data file by first encrypting the file then masking the encrypted version with some randomness generated from a Pseudo-Random Function (PRF) is being performed here[14].

3) Efficient Multi-Copy Provable Data Possession (EMC-PDP) schemes:

It's based on HLA's. In EMC-PDP models we resort to the diffusion property of any secure encryption scheme. Diffusion means that the output bits of the cipher text should depend on the input bits of the plain text in a very complex way. In an encryption scheme with strong diffusion property, if there is a change in one single bit of the plaintext, then the cipher text should completely change in an unpredictable way. This methodology of generating distinct copies is not only efficient, but also successful in solving the authorized users problem of the MRPDP

scheme to access the file copy received from the CSP. The two versions of the EMC-PDP schemes are[15]:

Deterministic EMC-PDP (DEMC-PDP) scheme
Probabilistic EMC-PDP (PEMC-PDP) scheme

In the DEMC-PDP version, the CSP has to access all the blocks of the data file, while in the PEMC-PDP, spot checking is performed by validating a random subset of the file blocks. It is a trade-off between the performance of the system and the strength of the guarantee provided by the CSP. In the PEMC-PDP scheme, we use the same indices for the challenged blocks across all copies. The rationale behind the PEMC-PDP scheme is that checking part of the file is much easier than the whole of it, and thus reducing the computation and storage overhead on the servers side.

4) Pairing based provable multi-copy data possession (PB-PMDP) scheme

This scheme provides an adequate guarantee that the CSP stores all copies that are agreed upon in the service contract, and these copies are intact. The authorized users can seamlessly access the copies received from the CSP. The PB-PMDP scheme supports public verifiability.

Generating unique differentiable copies of the data file is the core to design a multi-copy provable data possession scheme[16]. Identical data copies enable the CSP to simply deceive the owner by storing only one copy and pretending that it stores multiple copies. Using a simple yet efficient way, the proposed scheme generates distinct copies utilizing the diffusion property of any secure encryption scheme. There will be an unpredictable complete change in the ciphertext, if there is a single bit change in the plaintext. The interaction between the authorized users and the CSP is considered through this methodology of generating distinct copies, where the former can decrypt and access a file copy received from the CSP without recognizing the copy index. Homomorphic linear authenticators (HLAs) are basic building blocks in the proposed scheme.

5) Distributed and Replicated (DR-DPDP) scheme

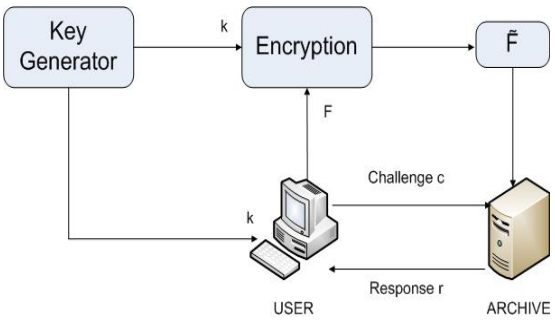
DR-DPDP is a scheme that provides transparent distribution and replication of user data over multiple servers. There are three entities in the model. The client, who stores data on the CSP, challenges the CSP to check the integrity of data, and updates the stored data[17]. The organizer, who is one of the servers in CSP and is responsible for communication with the client and other servers (acts as a gateway or load-balancer). The servers, who store the user data, perform provable updates on behalf of the client, and respond to the client challenges coming via the organizer. They only communicate with the organizer and there is no inter-server communication.

It is very important to observe that even though it seems like a central entity, the organizer is not expected to perform any disk operations or expensive group operations (e.g., exponentiation). He will only perform simple hashing, and work with a very small skip list. Hence, his load will be very light, making it very easy to replicate the organizer to prevent it from becoming a bottleneck or single-point-of-failure.

D. Static POR Schemes

1) Basic scheme:

Its a scheme which does not involve the encryption of the whole data. Only a few bits of data per data block are encrypted thus reducing the computational overhead on the clients. The client storage overhead is also minimized as it does not store any data with it. Hence this scheme suits well for thin clients [18].



It reduces the computational and storage overhead of the client as well as server. It also minimizes the size of the proof of data integrity so as to reduce the network bandwidth consumption.

2) POR for Large files:

Here the verifier stores only a single cryptographic key—irrespective of the size and number of the files whose retrievability it seeks to verify—as well as a small amount of dynamic state (some tens of bits) for each file. (One simple variant of this protocol allows for the storage of no dynamic state, but yields weaker security.) More strikingly, and somewhat counter intuitively, this scheme requires that the prover access only a small portion of a (large) file F in the course of a POR. In fact, the portion of F "touched" by the prover is essentially independent of the length of F and would, in a typical parameterization, include just hundreds or thousands of data blocks. Briefly, this POR protocol encrypts F and randomly embeds a set of randomly-valued check

blocks called *sentinels*.[19] The use of encryption here renders the sentinels indistinguishable from other file blocks. The verifier challenges the prover 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 F, then with high probability it will also have suppressed a number of sentinels. It is therefore unlikely to respond correctly to the verifier. To protect against corruption by the prover of a *small* portion of F, we also employ error-correcting codes.

3) Compact POR :

Here two new short, efficient homomorphic authenticators are being used. The first, based on PRFs, gives a proof-of-retrievability scheme secure in the standard model. The second, based on BLS signatures , gives a proofof-retrievability scheme with public verifiability secure in the random oracle model.[20]

4) General framework for POR :

A "spot-checking" mechanism is being used in the challenge-response protocol to detect adversarial behavior. In each challenge, a subset of file blocks is sampled, and the results of a computation over these blocks is returned to the client. The returned results are checked using some additional information embedded into the file at encoding time.[21]

5) HAIL:

HAIL manages file integrity and availability across a collection of servers or independent storage services. It makes use of PORs as building blocks by which storage resources can be tested and reallocated when failures are detected. HAIL does so in a way that transcends the basic single-server design of PORs and instead exploits both within-server redundancy and cross-server redundancy. HAIL relies on a single trusted verifier—e.g., a client or a service acting on behalf of a client—that interacts with servers to verify the integrity of stored files[22].

E. Dynamic POR

1) Data Correctness:

This is a data correctness scheme which involves the encryption of the few bits of data per data block thus reducing the computational overhead on the clients. Its based on the fact that high probability of security can be achieved by encrypting fewer bits instead of encrypting the whole data. The client storage overhead is also minimized as it does not store any data with it and it reduces bandwidth requirements. Hence this scheme suits well for small memory devices and low power devices. In this data integrity protocol the TPA needs to store only a single cryptographic key irrespective of the size of the data file F and two functions which generate a random sequence. The TPA does not store any data with it. The TPA before storing the file at the archive, preprocesses the file and appends some meta data to the file and stores at the archive. At the time of verification the TPA uses this meta data to verify the integrity of the data. Here the proof of data integrity protocol just checks the integrity of data. But the data can be stored, that is duplicated at redundant data centers to prevent the data loss from natural calamities. If the data has to be modified which involves updation, insertion and deletion of data at the client side, it requires an additional encryption of fewer data bits. So this scheme supports dynamic behavior of data.[23]

2) Public Auditability :

Solutions that meet various requirements such as high scheme efficiency, stateless verification, bounded use of queries and retrievability of data, etc. are required in a cloud computing environment. All schemes presented till now fall into two categories: private auditability and public auditability. Although schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. To ensure cloud data storage security, it is critical to enable a third party auditor (TPA) to evaluate the service quality from an objective and independent perspective. Public auditability also allows clients to delegate the integrity verification tasks to TPA while they themselves can be unreliable or not be able to commit necessary computation resources performing continuous verifications. Another major concern is how to construct verification protocols that can accommodate dynamic data files. In this paper, we explored the problem of providing simultaneous public auditability and data dynamics for remote data integrity check in Cloud Computing. Our construction is deliberately designed to meet these two important goals while efficiency being kept closely in mind. To achieve efficient data dynamics, we improve the existing proof of storage models by manipulating the classic Merkle Hash Tree (MHT) construction for block tag authentication.[24] To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

3) Public Verifiability:

This is a scheme with public verifiability: any TPA in possession of the public key can act as a verifier.

This protocol has the following features:

Public verification for storage correctness assurance: to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand.[25]

Dynamic data operation support: to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public verifiability and dynamic data operation support.

Blockless verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for both efficiency and security concerns.

Stateless verification: to eliminate the need for state information maintenance at the verifier side between audits throughout the long term of data storage.

4) A Dynamic PoR Scheme with O(logn) Complexity :

This scheme can be summarized as the following three stages:

Pre-process stage: Before outsourcing the file to the server, the client will preprocess the file and generate metadata. Then the client will outsource the file to the server and only keep the meta data[26].

Verification stage: The client will periodically check the integrity of its data. It will query the server randomly and ask the server to provide a proof. By verifying the proof with meta data, the client can detect the file corruption with high probability.

Update stage: The client will send the server a request to update the file. After each update, the server will prove to the client that the update is correctly executed.

5) E POR:

This is an efficient and secure POR scheme. In this scheme, a data block consists of s group elements and subsets of l blocks are accessed in each verification [27]. The storage overhead is 1=s of the data file size, and communication cost is O(1) bits per verification, and the computation cost is O(s) group exponentiations on the server side (prover) and O(`) group addition/multiplication/PRF on the client (verifier) side. It's proved that the proposed POR scheme is secure under a variant of Strong Diffie-Hellman Assumption.

IV. COMPARATIVE STUDY

This comparative study provides a consolidated report of all the techniques of the PDP schemes Static, Dynamic and multi-copy PDP Schemes in single and hybrid clouds and the various POR schemes.

Static PDP Schemes			
Scheme	**Technique used**	**Advantage**	**Disadvantage**
Basic	MAC	Secure and efficient	The communication complexity is linear with the queried data size which is impractical especially when the available bandwidth is limited.
PDI	Secure hash function	Checking part of the file is much easier than the whole of it.	1. Limited number of audits per file. 2. In each verification, the remote server has to do the exponentiation over the entire file. 3. Storage overhead on the verifier side.
Challenge-response	RSA Homomorphic hash function	The freshness of the response computation by the server is guaranteed by the fact that a challenge is never reused before reboot of the server.	Although the protocol does not require exponentiation of the entire file, a local copy of the fingerprints whose size is linear in the number of file blocks must be stored on the verifier side.
Data Storage Commitment	n-Power Computational Diffie-Hellman (n-PCDH) assumption	Makes use of storage space as large as the client's data	1. It only ensures that the server is storing something at least as large as the original data file but not necessarily the file itself. 2. In addition, the verifier's public key is about twice as large as the data file.
Privacy Preserving	Encryption	An external Third Party Auditor (TPA) can verify the integrity of files stored by a remote server without knowing any of the file contents.	1. The number of times a particular data item can be verified is limited and must be fixed beforehand. 2. Storage overhead on the TPA 3. Lack of support for stateless verification 4. Very high communication complexity
Database	Signature Aggregation	Signature aggregation enables bandwidth and computation efficient integrity verification of query replies.	1. Does not fulfill the completeness requirement. 2. Fails to provide block less verification
S-PDP & E-PDP	HVT'S, HLA'S KEA1 assumption	Client is convinced of data possession, without actually having to retrieve file blocks. Provide data format independence. Offers public verifiability.	1. HVTs are based on RSA and thus are relatively long. 2. The time taken to generate the tags is too long. 3. Since there is no indicator for the file identifier in the block tag, a malicious server can cheat by using blocks from different files if the data owner uses the same secret keys.
Dynamic PDP Schemes			
Scalable PDP	Based on cryptographic Hash function & symmetric-key encryption	1. Requires no bulk encryption of outsourced data and no data expansion due to additional sentinel blocks. 2. Supports secure and efficient dynamic operations on outsourced data blocks.	1. Number of updates and challenges is limited and fixed in advance. 2. It does not support block insertion operation
DPDP I	Rank-based authenticated skip list	1. Supports efficient authenticated operations on files at the block level, such as authenticated insert and delete. 2. Supports data possession guarantees of a hierarchical file system as well as file data.	It does not support efficient verification of the indices of the blocks, which are used as query and update parameters .
DPDP II	RSA trees	Blockless verification of data.	It has increased update time.
Dynamic PDP Schemes in Hybrid clouds			
Cooperative PDP	HVR, HIH, IPS, MPZKPS interactive proof system and multi-prover zero-knowledge proof system	1. Multi-prover zero-knowledge proof system (MP-ZKPS), which has completeness, knowledge soundness, and zero-knowledge properties. 2. It has security against data leakage attack and tag forgery attack.	Latency overhead and scalability of prototype has not been described.
Scalia	Multi- datacenter	High data durability and minimizes the storage cost for clients	Latency overhead and scalability of prototype has not been described.
Multi-Copy PDP Schemes(MC-PDP Schemes)			
BMC-PDP	Encryption	It generate unique distinguishable/differentiable copies of the data file.	1. The computation and communication complexities of the verification task grow linearly with the number of copies. 2. Key management is a severe problem.
MR-PDP	Signature aggregation	Each unique replica can be produced at the time of the challenge it can generate further replicas on demand	1. It does not address how the authorized users of the data owner can access the file copies from the cloud servers. 2. Computation overhead on both the verifier and server side. 3. Storage overhead
DEMC-PDP	Bilinear Map/Pairing.	1. Strongest guarantee at the expense of the storage overhead. 2. Shortest verification time	Storage and computation cost is larger.
PEMC-PDP	Bilinear Map/Pairing	Lowest storage overhead on the server side by using spot checking.	Storage and computation cost is larger.
PB-PMDP	BLS HLAs	1. It provides an evidence to the customers that all outsourced copies are actually stored and remain intact. 2. It allows authorized to seamlessly access the file copies stored by the CSP Supports public verifiability. 3. Secure against colluding servers.	While identifying corrupted copies the cost of extra storage, communication, and computation overheads occurs.
DR-DPDP	Rank-based authenticated skip list	It provides transparent distribution and replication of user data over multiple servers.	The computation time in the organizer becomes greater than that of the servers.

Static POR Schemes			
Scheme	**Technique used**	**Advantages**	**Disadvantages**
Basic	Encryption	Reduces the computational and storage overhead of the client as well as cloud storage server. It also minimizes the size of the proof of data integrity so as to reduce the network bandwidth consumption.	The number of queries that can be asked by the client is fixed apriori. But this number is quite large and can be sufficient if the period of data storage is short. It will be a challenge to increase the number of queries using this scheme.
POR for large files	Sentinel-based scheme	Ensures both possession and retrievability of files on archive service systems	Computationally cumbersome especially when the data to be encrypted is large. There will also be storage overhead at the server, partly due to the newly inserted sentinels and partly due to the error correcting codes that are inserted. Larger storage requirements on the prover.
Public verifiability scheme	Homomorphic authenticators & BLS signatures	Unlimited number of queries and requires less communication overhead	Used only for static data
2 phase protocol	Spot-checking	Lower storage overhead, tolerates higher error rates, and can be proven secure in a stronger adversarial setting.	Used only for static data
HAIL	MAC	Strong file-intactness assurance: Low overhead Strong adversarial model: Direct client-server communication:	Used only for static data
Dynamic POR Schemes			
Basic	Generation of metadata and its encryption	Reduces the computational and storage overhead of the client as well as the server. Minimizes the size of proof of data integrity so as to reduce the network bandwidth consumption.	Auditing multiple files from multiple clients simultaneously is not possible.
Public auditability	Bilinear aggregate signature and Merkle Hash Tree	Public auditability for storage correctness assurance Dynamic data operation support Blockless verification	Efficiency of the scheme remains unclear.
Public Verifiability	BLS, MHT	Public verification for storage correctness assurance Dynamic data operation support Blockless verification Stateless verification	Using the classic MHT construction will cause an efficiency problem.
CMBT Scheme	Cloud Merkle B+ tree , BLS signature	It is possible to detect file corruptions with high probability even if the CSP tries to hide them. Moreover, this scheme is able to support dynamic updates while keeps the same detection probability of file corruption. The worst case performance is when compared with other schemes is O (logn).	Less efficent
E-POR	Diffie-Hellman Assumption	Efficient and secure. It requires only a constant number of communication bits per verification.	-

REFERENCES

[1] K. Zeng, "Publicly verifiable remote data integrity," in ICICS, 2008, pp. 419–434.

[2] Deswarte, J-J. Quisquater, and A. Sa¨ıdane ,"Remote integrity checking," in 6th Working Conference on Integrity and Internal Control in Information Systems (IICIS), S. J. L. Strous, Ed., 2003, pp. 1–11.

[3] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," Cryptology ePrint Archive, Report 2006/150, 2006.

[4] P. Golle, S. Jarecki, and I. Mironov, "Cryptographic primitives enforcing communication and storage complexity," in FC'02: Proceedings of the 6th International Conference on Financial Cryptography, Berlin, Heidelberg, 2003, pp. 120–135.

[5] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems, Berkeley, CA, USA, 2007, pp. 1–6.

[6] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[7] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," Trans. Storage, vol. 2, no. 2, 2006.

[8] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol. 26, no. 1, 1983.

[9] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in ASIACRYPT '01: Proceedings of the 7th

International Conference on the Theory and application of Cryptology and Information Security, London, UK, 2001, pp. 514–532.

[10] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in CCS '07: Proceedings of the 14th ACM Conference on Computerand Communications Security, New York, NY, USA, 2007, pp. 598–609.

[11] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Secure Comm '08: Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, New York, NY, USA, 2008, pp. 1–10.

[12] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in ESORICS'09: Proceedings of the 14th European Conference on Research in Computer Security, Berlin, Heidelberg, 2009, pp. 355–370.

[13] Z. H. G.-J. A. H. H. Yan Zhu, Huaixi Wang and S. S. Yau, "Cooperative provable data possession," Cryptology ePrint Archive, Report 2010/234, 2010.A.Juels and B. S. Kaliski, "PORs: Proofs of Retrievability for large files," in CCS'07: Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007, pp. 584–597.

[14] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR- PDP: Multiple-Replica Provable DataPossession," in 28th IEEE ICDCS, 2008, pp. 411–420.

[15] Ayad F.Barsoum and M.Anwar Hasan Provable Possession and Replication of Data over Cloud Servers.

[16] Ayad F. Barsoum, M. Anwar Hasan. "Integrity Verification of Multiple Data Copies over Untrusted Cloud Servers"

[17] Mohammad Etemad and Alptekin Koc University, Istanbul, Turkey." Transparent, Distributed, and Replicated Dynamic Provable Data Possession".

[18] Sravan Kumar R, Ashutosh Saxena, Data Integrity Proofs in Cloud Storage, 978-1-4244-8953-4/11/@ 2011 IEEE.

[19] Ari Juels and Burton S. Kaliski, PORs: Proofs of Retrievability for Large Files, CCS '07 Proceedings of the 14th ACM conference on Computer and communications security, 978-1-59593-703-2,USA

[20] Hovav Shacham1 and Brent Waters, Compact Proofs of Retrievability, J. Pieprzyk (Ed.): ASIACRYPT 2008, LNCS 5350, pp. 90–107, 2008 International Association for Cryptologic Research 2008.

[21] Kevin D. Bowers, Ari Juels, Alina Oprea, Proofs of Retrievability: Theory and Implementation, CCSW'09, Journal of Systems and Software, v.85 n.5, p.1083-1095, May, 2012.

[22] Kevin D. Bowers, Ari Juels, Alina Oprea, HAIL: A High-Availability and Integrity Layer for Cloud Storage, ACM 978-1-60558-784-4 /09/11, USA.

[23] Malathi.M, Murugesan. T, A Scheme for Checking Data Correctness in the Cloud, 2012 International Conference on Information and Network Technology (ICINT 2012) IPCSIT vol. 37 (2012).

[24] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li, Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing, IEEE Trans. Parallel Distrib. Systems.

[25] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li, Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing, IEEE Trans. Parallel Distrib. Systems.

[26] Zhen Mo Yian Zhou Shigang Chen , A Dynamic Proof of Retrievability (PoR) Scheme with O(logn) Complexity, IEEE Trans. Parallel Distrib. Systems.

[27] Jia Xu and Ee-Chien Chang, Towards Efficient Proofs of Retrievability in Cloud Storage, IEEE Trans. Parallel Distrib. Systems