Provable Data Possession (PDP) and Proofs of Retrievability (POR) of Current Big User Data

Ieuan Walker^{1,2} · Chaminda Hewage¹ · Ambikesh Jayal³

Received: 5 October 2021 / Accepted: 11 November 2021 / Published online: 24 November 2021
© The Author(s) 2021

Abstract

A growing trend over the last few years is storage outsourcing, where the concept of third-party data warehousing has become more popular. This trend prompts several interesting privacy and security issues. One of the biggest concerns with third-party data storage providers is accountability. This article, critically reviews two schemas/algorithms that allow users to check the integrity and availability of their outsourced data on untrusted data stores (i.e., third-party data storages). The reviewed schemas are provable data possession (PDP) and proofs of retrievability (POR). Both are cryptographic protocols designed to provide clients the assurance that their data are secure on the untrusted data storages. Furthermore, a conceptual framework is proposed to mitigate the weaknesses of the current storage solutions.

Keywords Cloud security · Storage security · Provable data possession (PDP) · Proofs of retrievability (POR)

Introduction

Initial research into cloud storage technologies/solutions were aimed at providing authentication and integrity. They were mainly focussed on how to efficiently and securely return the complete and correct response to a client's query. 'Authenticity and Non-Repudiation of the answer to a database query' [10]. Then the focus is shifted to, how to query encrypted data efficiently and securely. In a study titled 'Secure Conjunctive Keyword Search over Encrypted Data' the authors researched how users can efficiently and securely retrieve encrypted documents using conjunctive keyword searches [16]. The proposed scheme in the above study provides the server a capability that allows the server to identify exactly those documents that matches conjunctive keywords".

With the wider usage of untrusted third-party data stores, it is important to ensure that data is not tampered with (ensure integrity) and data availability (it can recover the data with small file corruptions). This article, critically reviews two fundamental schemas/algorithms namely provable data possession (PDP) and proofs of retrievability (POR) that allow users to check the integrity and availability of their outsourced data on untrusted data stores (i.e., third-party data storages). The reviewed schemas are both cryptographic protocols designed to provide clients the assurance that their data are secure on the untrusted data storages. These two fundamental PDP and POR schemas do have different responsibilities. The main aim of PDP is to ensure that the client file is intact and has not been tampered, whereas POR main aim is to guarantee that the client can retrieve the file even with small file corruption. Both have real-world usage and is critical for today's data-centric world.

The main issue that these schemes attempted to solve is how to frequently, efficiently and securely verify that the client data are intact and able to retrieve the file if needed even if the file is corrupt. And also the ability to perform this verification without checking the entire file.

The differences between PDP and POR schemas are becoming insignificant with each new iteration or extensions to the original schemes reported in recent literature. This article demonstrates the differences and weaknesses in both

This article is part of the topical collection "Cyber Security and Privacy in Communication Networks" guest edited by Rajiv Misra, R K Shyamsunder, Alexiei Dingli, Natalie Denk, Omer Rana, Alexander Pfeiffer, Ashok Patel and Nishtha Kesswani.

✉ Chaminda Hewage
chewage@cardiffmet.ac.uk

¹ Cardiff Metropolitan University, Cardiff, UK

² Cardiff Council, Cardiff, UK

³ University of Canberra, Canberra, Australia

approaches and proposes a conceptual model as a building block for further research in this domain.

This research aims to identify the current algorithms in use in the cloud storage space. The two algorithms this article will critically analyse are provable data possession (PDP) and proofs of retrievability (POR). The reasons these algorithms were chosen are because they are closely related to each other and are critical in today's data-centric world.

This research consists of two main objectives:

1. Conduct a critical literature review on provable data possession (PDP) and proofs of retrievability (POR).
2. Based on the critical literature review conducted (based on the strength and weaknesses identified of modern cloud storages), this research then proposes a conceptual model.

This article is separated into three sections. The first section critically discusses provable data possession (PDP) and proofs of retrievability (POR) schemas, then the proposed conceptual model based on the previous findings is presented in section two, and finally the conclusion section summarises the findings.

Research Methodology

The research and data collection methodologies followed in this study are elaborated below. Inductive research approach is mainly used since the aim is to propose a conceptual framework based on the analysis of sample literature. The review is carried out using the publicly available, secondary data sources which discuss different aspects of validating and recovering data from an untrusted third-party storage provider. The main data sources used in this review are SCOPUS library, Web of Science (WoS) citation database, ACM library, IEEE Xplore, Google Scholar, Researchgate, etc. A number of keyword searches were used to find relevant studies and reviews necessary to answer the research questions of this article. The main keywords combinations included "Data integrity", "Cloud storage", "data retrievability", "validating data", and other relevant key words. An exclusion criterion was not used.

As for the schemas, they were selected for an analytical review based on the number of references found based on all the keyword combinations. In addition to the above keyword search by the authors, recommendations by previously published research, tutorials, surveys and reviews were used to select the schemas to focus on this review. The PDP and POR schemas have been analyzed, discussed and summarized. The academic papers from the literature for each schema were ordered chronologically, and where chosen due to the contribution they made to the overall schema,

and where found due to the amount of journal papers that referenced them.

Review of PDP and POR Schemes

Provable Data Possession (PDP)

Provable data possession (PDP) is a way to give the tenants a means to verify that their data, stored at untrusted storage is intact and has not been tampered with, without requiring the tenant to download the actual data.

A brief general overview of the PDP model is that the client pre-processes the data and sends it to an untrusted data store, while only keeping a small amount of metadata to use later. The client can then later ask the storage provider to prove that the data they sent has not been tampered with or deleted. All without requiring the file to be downloaded. [12]

Provable Data Possession Review

In 2007, authors in [17] presented a new type of scheme that they called 'Provable Data Possession' or PDP for short. The proposed scheme allows tenants who have stored files on an untrusted storage a means of verifying that their data is intact and has not been tampered with without requiring to download the actual file. This is very much important in current context where users store data in a number of third-party data storages (e.g. Google Drive, OneDrive, DropBox, etc.).

The main goal of this scheme is to be able to check the integrity of files as quickly as possible. It performs this using a minimal amount of metadata (which the tenants stores).

Following figures (Fig. 1 PDP pre-process and store diagram and Fig. 2 PDP verify server possession diagram) demonstrate how this schema works. Figure 1 PDP pre-process and store diagram illustrates the process of sending a file to the untrusted source, and Fig. 2 PDP verify server possession diagram shows how to verify the file on the server.

The pseudo code for preparing the file for uploading to the server is listed below.

1. Client breaks down the file into n blocks ($F_1, F_2, \dots, F_n$).
2. Client pre-process the file ($F'_1, F'_2, \dots, F'_n$) and generates metadata (M).
3. Metadata (M) is stored at the client side.
4. Client transfer the pre-processed file F' to the server.
5. Client deletes the local copy of the file.

The pseudo code for verification of the file listed below.

1. Client issues a random challenge (R) to the server to establish that the server has retained the file.

Fig. 1 PDP pre-process and store diagram **Fig. 2** PDP verify server possession diagram

2. The client requests that the server compute a function of the stored file.
3. Server sends back the response (P, the proof of possession) to the client.
4. Using its local metadata (M), the client verifies the response.

In the original paper [17], authors reviewed previous work on similar protocols but found that they had several drawbacks such as

- Require expensive service computation or communication over the entire file [14] and [9].
- Linear storage for the client.
- Do not provide security guarantees for data possession.

The main drawback to the PDP scheme proposed is that it only applies to static data. Which means if the client wishes to modify the data, they will have to run through the PDP scheme again from the start.

Ateniese, et al. has developed a dynamic PDP schema called Scalable PDP [3]. That allows somewhat limited dynamic data, meaning it enables appending, modifying and deleting blocks but does not allow inserting blocks.

The above scheme consists of two main phases namely setup and verification (also called challenge in the literature) [3], much like Guiseppe et al. schema [17]. But the new twist that the authors added to the PDP area is the idea of creating all future challenges during the setup phase and then store the pre-computed answers as metadata on the client [12]. “the owner (i.e., OWN) generates in advance t possible random challenges and the corresponding answers” [3]. Due to this approach, it limits the number of updates and challenges the client can perform. It also has a side effect of preventing the possibility of block insertions anywhere, and only allows the clients to append its blocks.

The authors recognise this limitation by stating “one potentially glaring drawback of our scheme is the prefixed (at setup time) number of verifications t ” [3]. They go on to say that the only way to increase the number of challenges and update would be by running through the setup phase again, requiring the client (OWN) to retrieve the entire file (D) from the server (SVR). But this approach would be problematic and impractical for large files.

Erway et al. [12] released a research paper in 2009 titled ‘Dynamic Provable Data Possession (DPDP)’ which was built upon the provable data possession model (PDP), which extends the functionality of it to support deletion,

modification and insertion of data. This is a significant improvement compared to prior work. Their work was released shortly after the proposed schema by Ateniese, et al. [3]. Both papers built upon the earlier work of Guisepppe et al. [17]. But the main difference between the scalable PDP [3] and the DPDP [12] is that Ateniese et al. [3] uses a random oracle model whereas DPDP scheme is “provably secure in the standard model” [12]. Authors go on to demonstrate the differences by creating a summary table comparing PDP [17], scalable PDP [3], DPDP I, DPDP II [12].

A number of improved PDP schemas have been reported in the recent literature [6, 19, 23, 26]. However, they all bare the same characteristics of original works as discussed above.

Discussion

Provable data possession is essential for being able to prove that the client's file (data) is intact and has not been tampered with, without requiring the tenant to download the actual data. Above is a detailed history of this schema starting with Guisepppe et al. [17]. Every update to the original work has its own benefits and drawbacks but with each variation, the PDP schemas have addressed most of the original concerns.

Guisepppe et al. [17] was a great starting point for the PDP schema and is still remain a viable choice at present for verifying archived data that does not need to be modified. However, in today's climate, many organisations store all their data on an untrusted server, so there is a need to verify dynamic data. This is where Ateniese et al. [3] and Erway et al. [12] extends Guisepppe et al. [17] adding the ability to efficiently verify dynamic data. Ateniese et al. [3] adds the ability to be able to modify and delete without needing to run through the entire schema again. But Ateniese et al. [3] still has some drawbacks that are discussed in this section. Erway et al. [12] goes a step further by also allowing block insertions, but they still suffer from small performance issues that authors go on to justify.

Proofs of Retrievability (POR)

Proofs of retrievability (POR), is very similar to the PDP schema. PDP demonstrates to a client that a server possesses the client's file and it has not been modified or deleted. POR allows the client to run an efficient audit protocol where the server proves that the client's file can be retrieved. POR schemes also have the ability to retrieve and fix files that has small file corruption with the use of error-correcting codes.

Proofs of Retrievability (POR) Review

The schema proposed by [20] is very similar to the provable data possession schema mentioned in the previous

Section (section “[Provable Data Possession \(PDP\)](#)”). The primary difference is that the proof of retrievability (POR) schemas focuses on the means for the client to receive proof that their data is begin stored without corruption and with the ability to retrieve the entire file even if the file has ‘small file corruptions’. This schema much like the first iteration of the PDP schema focuses on static storage and is designed for archived data.

The schema presented in [20] is relatively straightforward. It encrypts the file (F) and randomly adds check blocks which they have called ‘sentinels’. In this schema, the use of encryption renders the sentinels indistinguishable from other file blocks [20]. The client then challenges the storage provider on these sentinels. It does this by specifying the positions of a collection of sentinels and then asking the storage provider to return the associated sentinel values [20]. If the storage provider modified or deleted part of the file (F), there will be a high probability that it may also have suppressed several sentinels, and will be unlikely to respond with correct file blocks that correspond with the sentinels generated during the setup phase [20].

To protect against corruption, authors in Juels and Kaliski [20] employed error-correcting codes. This is to reveal small file corruptions that could be missed between sentinels. This means that the sentinels are used to detect if a large portion of the file has been modified or corrupted, and it would be unlikely to be able to retrieve or repair the file. If small parts of the file are corrupted, likely, this will not be detected but with the use of the error-correcting codes, the file will be retrieved and repaired. This depends on the strength of the error correction codes being deployed with the respective schemas [27].

Figure 3 demonstrates the setup process of the POR schema [15]. The main drawback of this process is the pre-processing/encoding of F required prior to storage [20]. The process of embedding sentinels and error-correcting codes imposes some computational overhead and cause larger storage requirements on the storage provider. Therefore, this will be problematic for large scale storage of data files.

As readers can see from the diagram above (Fig. 3), a file is split into blocks ($F_1 \dots F_n$) and error-correcting code (C) is added to each block. The resulting blocks are then encrypted using a block cypher. The final step generates the sentinels that are applied to the encrypted file.

The above steps are all executed on the client side before the file (F^*) is transferred to the server.

The sentinels are a small fraction of the encoded file, typically 2%, but the error-correcting codes imposes the bulk of the storage overhead” [20]. And for larger files the associated expansion factor $|F^*|/|F|$ can be fairly modest, e.g., 15% [20].

Fig. 3 POR setup process

Shacham and Waters [24] have proposed an improvement on Juels and Kaliski [20] schema called compact proofs of retrievability [24], but their solution is also for static data.

In their paper, authors explained and demonstrated two versions of their schema.

1. The first version is built from BLS signatures and secure in the random oracle model, which has the shortest query and response of any POR system [24].
2. And the second version builds elegantly on pseudorandom functions (PRFs) and is secure in the standard model, which has the shortest response of any POR system [24].

Both are based on a homomorphic authenticator for file block, which essentially means that block integrity values that can be efficiently aggregated to reduce bandwidth in a POR protocol [5]. Juels and Kaliski [20] scheme uses MAC-Based message authentication, which according to Bowers et al. [5] would increase the size of the response. According to Bowers et al. [5], if each authenticator is λ bits long, as required in the Juels–Kaliski model, then the size of the response is $\lambda \cdot (s + 1)$ bits, where the ratio of file block to authenticator length is s : 1.9. Here s is the size of the file.

The use of homomorphic authenticators rather than MAC-based message improves the response length. Homomorphic authenticators are explained in more detail by Krohn et al. [21], who stated: “It is fast to compute, efficiently verified using probabilistic batch verification, and has provable security under the discrete-log assumption”.

The main advantages of this schema [24] over Juels and Kaliski [20] are the smaller response length. Furthermore,

unlike the Juels and Kaliski [20] schema, users are not limited in the number of verification they can perform. However, Shacham and Waters [24] still have the same drawback and that is, it only works for static file archival and you are not able to update or modify the file without removing the original file and re-uploading.

Bowers et al. [5] scheme improves on the work done by Juels and Kaliski [20] and Shacham and Waters [24]. Moreover, this schema is a variant on the Juels and Kaliski [20] POR scheme, and researchers used it as a starting point. Bowers et al. [5] have improved on two key parts of a POR system and that is,

- Allowing for higher acceptance of error rate, while still being able to retrieve the original file.
- Lower data overhead on the uploaded file.

The error-correcting method that Bowers et al. [5] employed is different from Juels and Kaliski [20] and Shacham and Waters [24]. It used an inner and outer error-correcting code which allows a higher error tolerance rate. Bowers, et al. [5] described the inner and outer code as “The two codes play complementary roles, but operate in distinct ways and at different protocol layers”. The inner code is computed on the fly by the server; therefore, it does not create a storage overhead. But it does impose a computational burden on the server when it responds to client challenges. This is because the server must retrieve the selected blocks from the challenge and apply the inner code each time.

The outer code has a similar effect to the error-correcting code in Juels and Kaliski [20] and Shacham and Waters [24] schemas where it has an insignificant effect on the servers computational power, but does increase the stored file size, therefore, the outer error-correcting code is embedded with the file.

Guo et al. [18] has modified the POR schema to not only focusing on data integrity, but data availability when there is a server failure. Researchers achieved this by utilising the clouds ability to replicate the data for redundancy. They then adapted the POR schema to ensure that if some of replicas are corrupted, the file can still be restored by means of the healthy replicas [18]. To achieve this, they needed to prove that multiple replicas of the file are indeed stored [18].

Authors started by identifying existing solutions for this problem such as

- Multiple-replica provable data possession [7, 8].
- Transparent, distributed, and replicated dynamic provable data possession [13].
- Provable multicopy dynamic data possession in cloud computing systems [4].

- MuR-DPA: Top-down levelled multi-replica Merkle Hash tree-based secure public auditing for dynamic big data storage on cloud [22].

Based on the review of prior work in the domain, authors came to the conclusion that all proposed methods solved separate and specific issues but still inadequate for wider use. Authors then discussed a viable solution called ‘Mirror’ [2]; however, they identified numerous security flaws with the implementation.

Discussion

Proofs of retrievability (POR) are essential to proving that a client can still retrieve the entire file without corruption. This section analysed a number for different POR approaches starting with the pioneering work proposed by Juels and Kaliski [20]. Much like the start of the PDP schema, they too focused on static data, meaning that this schema does not support updates. Another drawback of this schema is the number of queries the clients can make is fixed, which adds a restriction on the lifetime of the scheme.

Then the research (compact proofs of retrievability) proposed by Shacham and Waters [24] advanced the work of Juels and Kaliski [20] schema but it too only supported static data. The enhancements proposed by Shacham & Waters [24] over Juels and Kaliski [20] are two folds, namely,

- Smaller response improved the bandwidth usage.
- The proposed schema is not limited by the number of verification users can perform.

The scheme proposed by Bowers et al. [5] is based on the works of Juels and Kaliski [20] and Shacham and Waters [24] and improves two key parts of the POR system, namely,

- Allowing for higher acceptance of error rate, while still being able to retrieve the original file.
- Low data overhead on the uploaded file.

But the POR schemas mentioned above all share the same drawback, and that is they only support static files. However, recent literature has some evidence of dynamic POR approaches [11, 25] even though they bare the whole mark basic design of the original POR schemes as discussed in this article.

The Proposed Conceptual Model

This section presents the proposed conceptual model based on the research and review presented in the above sections. To create this conceptual model, both strengths and weaknesses of both schemas are evaluated. Table 1 Provable Data Possession (strengths and weaknesses) and Table 2 Proofs of Retrievability (strengths and weaknesses) identify the strengths and weaknesses of each approach.

Both models are very similar to each other

- Both rely on metadata being stored on the client.
- Both pre-processes the file on the client.
- Both attempts to limit the size of bandwidth used.

Table 1 Provable data possession (strengths and weaknesses)

Strength	Weaknesses
Proves that a file is intact and has not been tampered with	You have to decide between flexibility and performance Currently adding the ability of block insertions decreases the performance
Does not require downloading the entire file	Does not guarantee that the client can retrieve the file
In the later approaches, the schema is more flexible allowing—appending, modifying, inserting, deleting entire blocks, without needing to run through the entire process again	
Good use of bandwidth	

Table 2 Proofs of retrievability (strengths and weaknesses)

Strength	Weaknesses
Proves that the file is retrievable (without corruption)	Not flexible. Most of the current schemas only work with static data
Fix files with small file corruptions	The number of queries the clients can make is fixed, which puts a restriction on the lifetime of the scheme
Good use of bandwidth	Data expansion due to additional sentinel blocks
Does not require downloading the entire file	

- Both attempts decrease latency and time taking to perform the checks.

The differences lie within the goal of each approach, which are

- PDP—proves that a file is intact and has not been tampered with
- POR—proves that the file is retrievable (without corruption).

The conceptual model proposed in this article is a combination of both models with the end goal of the model to be able to prove that the file is intact and retrievable. This model is based on the PDP model created by Erway et al. [12]. The model processes are built on the PDP proposed by Erway et al. [12] due to it being the most advanced model reported in the literature which has the fewest limitations compared to other schemas discussed in this article.

The decision was made to base the conceptual model on the PDP model over the POR model. Because the complexity of the PDR model will hinder the simple implementation. Therefore, authors decided to implement POR model into the PDP model rather PDP into the POR model.

Below are two key aspects to the POR model that differ from the PDP model.

1. Blocks which they have called ‘sentinels’.
2. Error-correcting code.

The first is the use of check blocks ‘sentinels’. These are blocks of data used to challenge the server at a later date. Sentinels are indistinguishable from other file blocks

and the server will be asked to return specific file blocks to prove that the file is retrievable.

Then there are error-correcting codes, these are created to protect against corruption Juels and Kaliski [20] and are used to reveal small file corruptions that could be missed between sentinels.

This means that the sentinels are used to detect if a large portion of the file has been modified or corrupted, and it would be unlikely to be able to retrieve or repair the file. If small parts of the file are corrupted, likely, this will not be detected but with the use of the error-correcting codes, the file will be retrieved and repaired.

If both of these aspects can be merged into the PDP model then the model would have the benefits of both the PDP and POR models. Figures 4 and 5 illustrate the proposed conceptual model. Much like the other PDP and POR schemas, there are two schematic diagrams, first to show the pre-processing and upload of the file and the other to query the server for the proofs (possession and retrievability).

As can be seen, the diagram are remarkably similar to the PDP diagrams shown earlier in this article. The main difference between the two is the pre-processing of the file on the client to embed more information into the file and to store slightly more metadata to be able to benefit from both approaches.

The pseudo code for preparing the file for uploading to the server is listed below.

1. The file is split into blocks ($F_1 \dots F_n$).
2. Error-correcting code (C) is added to each block.
3. The resulting blocks are then encrypted using a block cypher ($F'_1, F'_2, \dots$).

Fig. 4 File upload

Fig. 5 Retrieve proof

4. The sentinels are generated for the encrypted file ($F^*_1, F^*_2, \dots$).
5. The metadata (M) is generated using the sentinels (i.e., $F^*_1, F^*_2, \dots$) and store the M at the client side.
6. The above steps are all executed on the client-side before the file (F^*) is transferred to the server.
7. The file is deleted from the client side.

The issues this research foresees is that more research will be needed to make the POR model compatible with dynamic data before the model can be implemented. At this current point of time attempting to merge both approaches would limit the model to static data only. An article by Curtmola et al. [7, 8] added the error-correcting codes to the PDP schema, but their research was based on the original PDP schema which was only compatible with static data. The other issue is around server overhead with both the PDP and POR process adding more data to the file it would increase the server overhead and bandwidth.

A recent article published by Anthoine et al. [1] has fixed one of the main issues identified above (i.e., the POR model compatible with dynamic data). Further analysis is needed to investigate this new POR approach and identify the strengths and weaknesses of their approach. In future research, if the new schemes such as the one reported in Anthoine et al. [1] have fixed the dynamic data issue with the POR schema, authors would investigate the possibility of using the POR schema and combining it with the PDP schema the above conceptual model is based on.

Conclusion

PDP and POR schemas do have different responsibilities. PDP main aim is to ensure that the client file is intact and has not been tampered, whereas POR main aim is to guarantee that the client can retrieve the file even with small file corruption. Both have real-world usage and is critical for today's data-centric world.

Merging both of these schema could have significant benefits. Hence, a conceptual model is proposed in this article after analysing the pros and cons of existing schemas from the relevant research literature. A previous work called 'Robust remote data checking' [7, 8] proved that this is a possibility. However, these approaches were developed quite early in the development of both schema which means that it does not support dynamic data, as at the time neither of the schema did.

More research is needed to make the POR model compatible with dynamic data before the model can be implemented. At this current point of time attempting to merge both approaches would limit the model to static data only, which would have little benefit over the previous research.

Once POR does support dynamic data attempting to merge both schemas should be successful, but this research does foresee an issue with server overhead with both the PDP and POR process adding more data to the file it would increase the server overhead and bandwidth, so further research will be needed to make the process more efficient. Even though the proposed conceptual model provides a

basic framework, further research and experimentation will be carried out to implement and test the proposed scheme and also to compare its performance with other state of the art mechanisms.

Funding Funded by Knowledge Economy Skills Scholarships (KESS 2) supported by European Social Funds (ESF) through the Welsh Government and Ultranyx Ltd.

Declarations

Conflict of interest All of the authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Anthoine G, Jean-Guillaume D, Michael H, Mélanie de J, Aude M, Clément P, Daniel R. Dynamic proofs of retrievability with low server storage. *CoRR*. 2021. [arXiv: 2007.12556](https://arxiv.org/abs/2007.12556).
2. Armknecht F, Barman L, Bohli J, Karame GO. Mirror: enabling proofs of data replication and retrievability in the cloud. *USENIX Security Symposium*. 2016.
3. Ateniese G, Pietro RD, Mancini LV, Tsudik G. Scalable and efficient provable data possession. *Istanbul: ACM*; 2008. p. 91–910.
4. Barsoum AF, Hasan AM. Provable multicopy dynamic data possession in cloud computing systems. *IEEE Trans Inf Forensics Secur*. 2014;10(3):485–97.
5. Bowers KD, Juels A, Oprea A. Proofs of retrievability: theory and implementation. *Chicago: ACM*; 2009. p. 43–54.
6. Chen R, Li Y, Yu Y, Li H, Chen X, Susilo W. Blockchain-based dynamic provable data possession for smart cities. *IEEE Internet Things J*. 2020;7(5):4143–54. <https://doi.org/10.1109/JIOT.2019.2963789>.
7. Curtmola R, Khan O, Burns R. Robust remote data checking. In: *Proceedings of the 4th ACM international workshop on storage security and survivability*. Alexandria: Virginia, USA; 2008. p. 63–68.
8. Curtmola R, Khan O, Burns R, Ateniese G. Multiple-replica provable data possession. *Beijing: IEEE*; 2008. p. 411–20.
9. Deswarte Y, Quisquater J-J, Sadane A. Remote integrity checking. In: *Jajodia S, Strous L, editors. Integrity and internal control in information systems*. Boston: Springer; 2004. p. 1–11.
10. Devanbu P, Gertz M, Martel C, Stubblebine SG. Authentic third-party data publication. *Boston: Springer*; 2002. p. 101–12.
11. Elaine S, Stefanov E, Papamanthou C. Practical dynamic proofs of retrievability. *Proc ACM SIGSAC Conf Comput Commun Secur*. 2013. <https://doi.org/10.1145/2508859.2516669>.
12. Erway C, Kùpçü A, Papamanthou C, Tamassia R. Dynamic provable data possession. *ACM Trans Inf Syst Secur*. 2015;17(4):29. <https://doi.org/10.1145/2699909>.
13. Etemad M, Kupcu A. Transparent, distributed, and replicated dynamic provable data possession. *Berlin: Springer-Verlag*; 2013. p. 1–18.
14. Filho DLG, Barreto PSIM. Demonstrating data possession and uncheatable data transfer. *IACR Cryptol*. 2006;2006:150.
15. Gilberg OR. Cloud security without trust. *Norwegian: Norwegian University of Science and Technology Department of Telematics*; 2014.
16. Golle P, Staddon J, Waters B. Secure conjunctive keyword search over encrypted data. *Berlin: Springer*; 2004. p. 31–45.
17. Guiseppe A, et al. Provable data possession at untrusted stores. *Virginia: ACM Alexandria*; 2007. p. 598–609.
18. Guo W, et al. Improved proofs of retrievability and replication for data availability in cloud storage. *Comput J*. 2020;63(8):1216–30.
19. Ieuan W. Provable Data Possession (PDP) and Proofs of Retrievability (POR) of current big user data. *MPhil Thesis, Cardiff Metropolitan University, UK*. 2021.
20. Juels A, Kaliski B. PORs: Proofs of Retrievability for large files. In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 584–97.
21. Krohn MN, Freedman MJ, Mazieres D. On-the-fly verification of rateless erasure codes. *Berkeley: IEEE*; 2004.
22. Liu J, Tong J, Mao R, Bohn J, Messina L, Badger D. Leaf, NIST-Cloud computing reference architecture NIST Special Publication; 2011. p. 500–292.
23. Liu Y, et al. New provable data transfer from provable data possession and deletion for secure cloud storage. *Int J Distrib Sens Netw*. 2019. <https://doi.org/10.1177/1550147719842493>.
24. Shacham H, Waters B. Compact Proofs of Retrievability. *J Cryptol*. 2008;26(3):442–83.
25. Wei G, Hua Z, Sujuan Q, Fei G, Zhengping J, Wenmin L, Qiaoyan W. Improved Proofs of Retrievability and replication for data availability in cloud storage. *Comput J*. 2020;63(8):1216–30. <https://doi.org/10.1093/comjnl/bxz151>.
26. Wei G, Hua Z, Sujuan Q, Fei G, Zhengping J, Wenmin L, Qiaoyan W. Outsourced dynamic provable data possession with batch update for secure cloud storage. *Futur Gener Comput Syst*. 2019;95:309–22. <https://doi.org/10.1016/j.future.2019.01.009>.
27. Wang R. Research on data security technology based on cloud storage. *Proced Eng*. 2017;174:1340–55. <https://doi.org/10.1016/j.proeng.2017.01.286>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.