ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

FORMAL VERIFICATION OF DATA MODIFICATIONS IN CLOUD BLOCK STORAGE BASED ON SEPARATION LOGIC

SHAGAM GEETHA[1], A. REETHIKA REDDY[2], D. MADHU[3], B. BHARATH[4], MA AZEEM[5]

[1]Assistant Professor, Dept. of CSE, Guru Nanak Institutions Technical Campus, Ibrahimpatnam, Telangana, India

[2, 3, 4, 5] B.Tech, Dept. of CSE, Guru Nanak Institutions Technical Campus, Ibrahimpatnam, Telangana, India

ABSTRACT— It is a famous type of cloud storage. It has the closest architecture to the underlying storage and can provide interfaces for other types. Data modifications in CBS have potential risks such as null reference or data loss. Formal verification of these operations can improve the reliability of CBS to some extent. Although separation logic is a mainstream approach to verifying program correctness, the complex architecture of CBS creates some challenges for verifications. This paper develops a proof system based on separation logic for verifying the CBS data modifications. The proof system can represent the CBS architecture, describe the properties of the CBS system state, and specify the behaviour of CBS data modifications. Using the interactive verification approach from Coq, the proof system is implemented as a verification tool. With this tool, the paper builds machine-checked proofs for the functional correctness of CBS data modifications. This work can thus analyse the reliability of cloud storage from a formal perspective.

Index Terms – Cloud Block Storage (CBS), cloud storage, formal verification, separation logic, Coq, program verification, data modification, functional correctness, interactive theorem proving, cloud reliability, formal methods, cloud computing.

I. INTRODUCTION

In recent years, with the rapid development of Internet technology, cloud services have

made breakthrough progress. However, with more frequent use, the reliability of cloud storage has been questioned in recent years. In the United States, nearly USD 285 million is lost per year directly caused by cloud storage failures [1]. For example, in 2020, dozens of servers in Amazon's cloud services have crashed for five hours simultaneously, which was caused by a "small increase in storage capacity" [2]. Although many providers deploy advanced failover methods, cloud storage still suffers from many service failures and reliability problems. Therefore, it is necessary to improve the reliability of cloud storage, to avoid the recurrence of such accidents and reduce the risk of data loss.

The reliability involves two main dimensions: the stability of hardware infrastructure and the correctness of data management programs [3]. The management programs are directly related to users. Improving their correctness can thus improve the reliability of cloud storage. Currently, the mainstream cloud storage in the market can be classified into three categories: file storage, object storage, and block storage [4]. This classification is about the different system architectures and data storage levels.

Cloud block storage (CBS) has the most direct interaction with the storage medium. It can even directly provide interfaces for the other two types [5]. Hadoop distributed file system (HDFS) [6] is one of the most representative CBS products. HDFS has a typical CBS storage approach, which allows for deploying large data sets by dividing them into discrete blocks. HDFS has a master-slave storage architecture. The master node maintains a file system, and the slave nodes store the block's actual content. The other CBS products have a similar architecture and storage method to HDFS. Generally speaking, the management of data in CBS involves a logical storage relation like "file-block content". To analyse and verify the correctness of programs in CBS, we need to pay attention to this storage relation. Early versions of CBS products, including HDFS, follow the simple coherency model [7], which means that once a file is created, users cannot modify the file's content. That model is designed to simplify data coherency issues and enable high throughput data access. However, with the growing popularity of CBS, various usage scenarios require CBS to support the modification of existing files. Therefore, the updated CBS products started to support data modifications. HDFS typically introduces two management programs: append and truncate, to add and remove content at the end of a file. Compared

to basic CBS data operations, these modifications are more complex to execute and more likely to cause logical storage errors, such as null reference or block loss. Therefore, it is necessary to analyse their correctness. The correctness of a program generally refers to the fact that each input will produce the desired result. There are two main approaches to verifying it: testing and formal verification. The testing approach is difficult to achieve overall coverage for large systems [8]. Formal verification models complex systems as mathematical entities. It can mathematically analyse whether a program is correct [9]. Theorem proving is a formal verification technique based on mathematical logic [10]. It models a computer system as an axiomatic system with reasoning capabilities. In recent years, theorem proving has produced many results for the verification of traditional storage. But cloud storage has a more complex architecture than local storage. In addition, the size of cloud storage may change as the requirements grow. Such factors make formal verification of cloud storage challenging, and it is also hard to directly apply the verification techniques for traditional storage. Therefore, researchers need to extend and innovate on the existing approaches. The current validation works on cloud storage mainly focus on data consistency, high availability, or integrity. However, there are fewer discussions of data operations from a logical storage perspective. Logical storage involves information to locate the target data, such as file directories and block locations. We advocate using the theorem proving technique to analyse and verify the correctness of CBS data operations, especially modifications, as a way to reduce the errors of logical storage.

Separation logic [11], [12] is a breakthrough in the area of program verification. It was originally proposed by O'Hearn and Reynolds, as a formal proof system based on Hoare logic [13]. Separation logic is an intuitive way to verify programs with explicit memory management [14].

It describes the behaviour of a program by a triple as "", such a triple is called a specification of.

The precondition is a predicate that describes the input state, and the postcondition describes the output state. Separation logic has two features: small-footprint specification and local reasoning. According to these two features, one may verify a program by focusing on only

the partial state relevant to execution without discussing the irrelevant state [15]. Thus,

separation logic is suitable for verifying complex storage systems, especially for CBS. To improve the practicality and efficiency of theoretical models, researchers usually implement a verification tool to facilitate reasoning. Coq [16] is one of the mainstream implementation environments. The proof system developed in Coq can be used as a verification tool, and the proof process of verifying a program in such a tool retains mathematical rigor. Using that tool, one can build machine-checkable proof and avoid possible negligence's in pen-and-paper inference [17]. Hence, many practical tools are used for analysing computer systems, from data structures to operating system kernels.

II. RELATED WORK

A. A tool for verifying cloud block storage based on separation logic

Cloud storage is now widely used in production and people's life. Verifying the correctness of hypervisors in cloud storage can effectively improve the reliability of the whole system. Cloud block storage (CBS) has the closest storage architecture to the bottom layer. In this study, a tool is implemented for analyzing and verifying the correctness of hypervisors in CBS, by using the interactive theorem prover Coq. Based on separation logic, the implementation of the proof system in the tool mainly consists: First, a modeling language is defined to abstract the CBS into a two-tier structure, and to formally represent the CBS state and the hypervisor; second, several predicates are defined to describe the state properties of the CBS, and the logical relationships between predicates are illustrated; finally, a separation logic triple for CBS is defined to describe the behavior of a program, and the reasoning rules required to verify the triples are stated. In addition, several proof examples are introduced in this study, to present the tool's ability to represent and reason about the real hypervisor in CBS.

B. An adaptation-complete proof system for local reasoning about cloud storage systems

The rapid growth of data presents a significant challenge to the capability of traditional storage technologies to collect and manage data. Cloud storage systems (CSSs) have been proposed as a method to improve storage capacity. To safely and effectively manage cloud storage data and improve data service quality, it is necessary to verify the correctness of CSS management programs. However, the complexity of these systems renders program verification difficult. In this paper, we propose a Hoare-style proof

system, in conjunction with two languages, to analyze and verify CSS management programs. The first is a modeling language that describes the program execution. The second is an assertion language based on Separation Logic (SL), used to describe the properties of the CSS file-block-location storage structure. The proof system supports modular local reasoning for CSS programs by a set of adaptation rules, which enable the condition of specifications to be applied to broader contexts. A key question that arises is whether the proof system can meet adaptation completeness. If so, arbitrary satisfiable specifications can be adjusted using the adaptation rules. To this end, we developed local predicate transformers and used their domain to interpret all types of commands. By finding the smallest local predicate transformer, we established adaptation completeness. In summary, this work provides a formalization of automatic modular reasoning patterns and lays a theoretical foundation for the compositional program verification of CSS.

C. Separation logic for sequential programs

This paper presents a simple mechanized formalization of Separation Logic for sequential programs. This formalization is aimed for teaching the ideas of Separation Logic, including its soundness proof and its recent enhancements. The formalization serves as support for a course that follows the style of the successful Software Foundations series, with all the statement and proofs formalized in Coq. This course only assumes basic knowledge of λ-calculus, semantics and logics, and therefore should be accessible to a broad audience.

D. An efficient leakage-resilient authenticated key exchange protocol suitable for IoT devices

Authenticated key exchange (AKE) protocol for client–server environments are a significant cryptographic primitive that provides communication confidentiality and mutual authentication between clients and servers.

In an Internet of Things (IoT) environment, clients typically employ IoT devices with limited computing capability to interact with servers through the Internet. Numerous AKE protocols suitable for IoT devices, called AKE-IoT protocols, have been proposed. Recently, side-channel attacks have been conducted to defeat traditional cryptographic protocols because a side-channel adversary can retrieve partial content of long-term or short-term secret keys.

Several leakage-resilient AKE (LRAKE) protocols were presented to counteract such attacks. Unfortunately, the existing LRAKE protocols are not suitable for IoT devices because expensive pairing operations are required for client sides. In this article, we propose the first efficient LRAKE protocol suitable for IoT devices, named LRAKE-IoT. By the unbalanced computation method, no pairing operation is required for client sides in our protocol.

In the generic bilinear pairing group model, security analysis is conducted to demonstrate the security of the proposed protocol in the continuous-leakage-resilient extended-Canetti–Krawczyk model. Finally, computational experiences on two IoT devices are given to show that the proposed protocol is well-suited for IoT devices.

III. PROPOSED METHODOLOGY

The management programs are directly related to users. Improving their correctness can thus improve the reliability of cloud storage. Currently, the mainstream cloud storage in the market can be classified into three categories: file storage, object storage, and block storage.

The current validation works on cloud storage mainly focus on data consistency, high availability, or integrity. However, there are fewer discussions of data operations from a logical storage perspective.

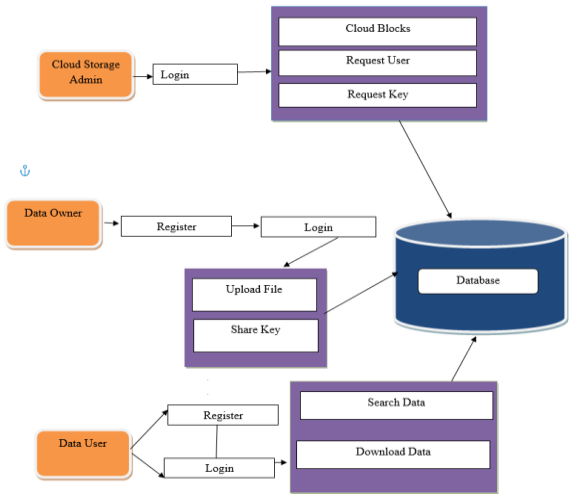
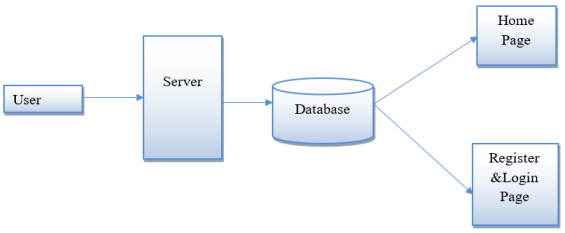


Fig. 1: System Overview

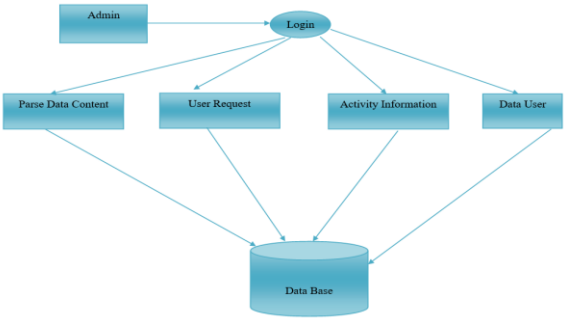
1. User Interface Design

In this module we design the windows for the project. These windows are used for secure login for all users. To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page.



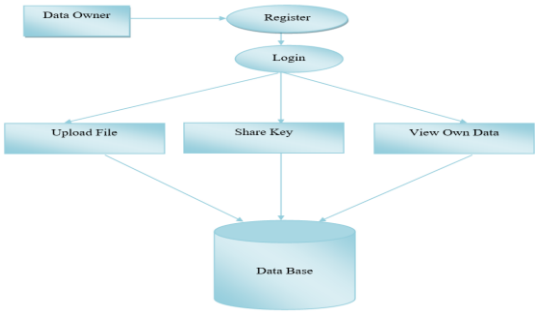
2. Admin

This is the first module a admin. Admin can login from a user id and password. Admin can also a parse data content. The admin can also have a user request to add a users. Admin can also have a activity information all recorded information.



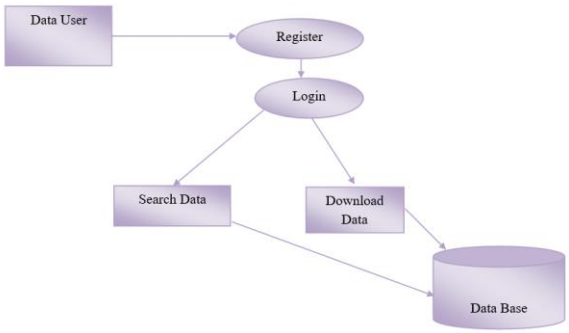
3. Data Owner

This is the third module of this project. In this project data owner has a register with all details and login with a user id and password. The Data owner has a upload a file the file has a upload with a text file to the data. The data owner has a share a key to the user. The data owner has a view own data to store in the database.



4. Data User

This is the fourth module of this project. Data user has a register with all details and then login with a email id and password. Data user has a search the data owner has a store we can search a data. Data User has a key verification the keys are verify the correct then it will have a decrypt a file download.



IV. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results show that the proposed separation logic-based proof system successfully verifies the correctness of Cloud Block Storage (CBS) data modification operations. The implementation

using the Coq interactive theorem prover generates machine-checked proofs, ensuring reliable verification of data updates and reducing the risk of data loss or invalid references. The framework accurately models CBS architecture and validates system behavior through formal verification.



Fig. 2: Home Page

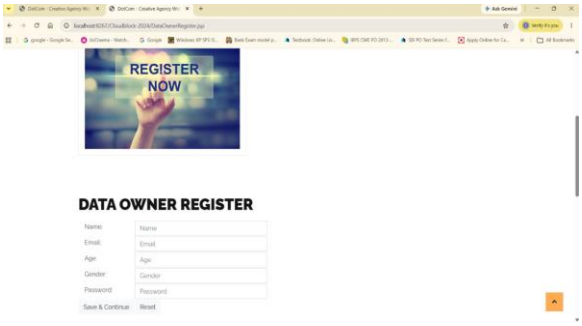


Fig. 3: Registration

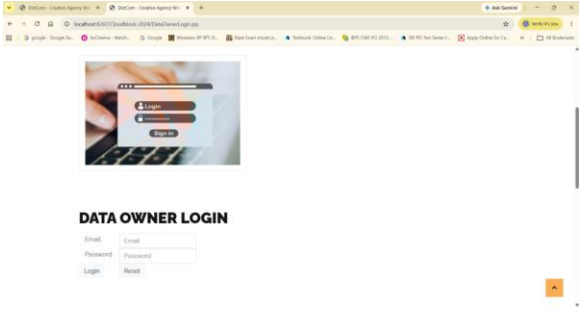


Fig. 4: Login

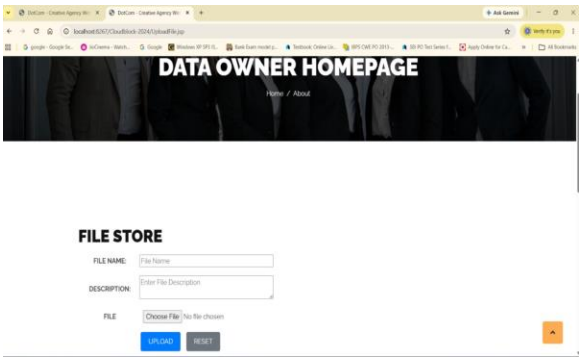


Fig. 5: Upload File

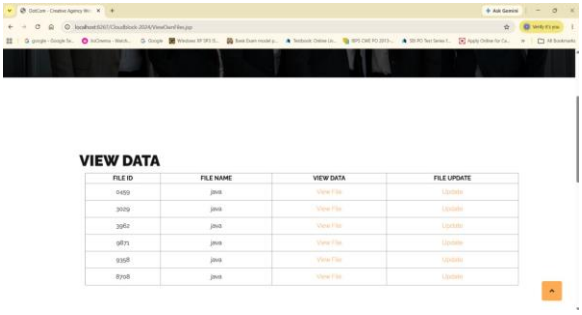


Fig. 6: View Data

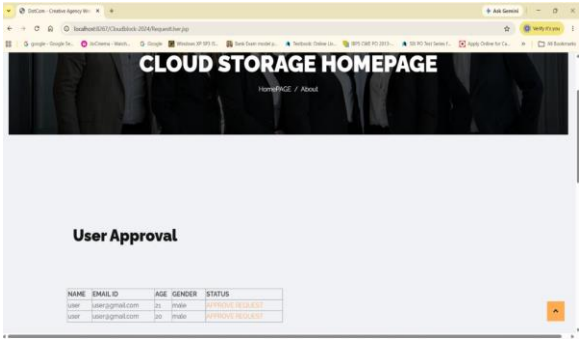


Fig. 7: User Approval

V. CONCLUSION

To improve the reliability of cloud block storage, this paper focuses on verifying the correctness of its data modifications. A proof system is developed based on separation logic, and it is subsequently implemented as a verification tool in Coq. The construction of

the proof system is explicit and specific, which can enhance the rigor of the corresponding verifications in Coq. This paper illustrates that our work can represent, specify, and verify the actual CBS management programs. In particular, this paper builds machine-checked proofs for the functional correctness of CBS data modifications.

REFERENCES

[1] L. Tung, "Amazon: here's what caused the major AWS outage last week," Available at: https://www.zdnet.com/article/ amazon-heres-what-caused-major-aws-outage-last-week-apologies/, 2020.

[2] A. Gawanmeh and A. Alomari, "Challenges in formal methods for testing and verification of cloud computing systems," Scalable Computing:Practice and Experience, vol. 16, no. 3, pp. 321–332, 2015.

[3] IBM Cloud Education, "What is block storage?," Available at: https://www.ibm.com/cloud/learn/block-storage, 2019.

[4] M. Mesnier, G. R. Ganger, and E. Riedel, "Object-based storage," IEEE Communications Magazine, vol. 41, no. 8, pp. 84–90, 2003.

[5] K. Shvachko, H. R. Kuang, S. Radia, et al., "The hadoop distributed file system," in Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies, Incline Village, NV, USA, pp. 1–10, 2010.

[6] Apache, "HDFS architecture guide," Available at: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, 2020.

[7] C. Newcombe, T. Rath, F. Zhang, et al., "How Amazon web services uses formal methods," Communications of the ACM, vol. 58, no. 4, pp. 66–73, 2015.

[8] J. M. Wing, "A specifier's introduction to formal methods," Computer, vol. 23, no. 9, pp. 8–22, 1990.

[9] J. H. Gallier, Logic for Computer Science: Foundations of Automatic Theorem Proving, 2nd ed., Dover Publications, New York, NY, USA, pp. 1–12, 2015.

AUTHORS Profile



Mrs. SHAGAM GEETHA has received her B.Tech in 2012 from Nagarjuna Institute of Technology, JNTUH. She completed her M.Tech in Computer Science and Engineering (CSE) in 2014 from Vidya Vikas Institute of Technology, JNTUH.



Mrs. A. REETHIKA REDDY pursuing B.Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.



Mr. D. MADHU pursuing B.Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.



Mr. B. BHARATH pursuing B.Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.



Mr. MA AZEEM pursuing B.Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.