IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, June 2025
Impact Factor: 7.67

Formal Verification of Data Modifications in Cloud Block Storage Based on Separation Logic

M. Ajaykumar[1], G. Pranay kumar[2], CH. Mohan Sai[3], A. Shedrak[4]
Assistant Professor, Department of CSE[1]
Students, Department of CSE[2,3,4]
Guru Nanak Institute of Technology, Hyderabad, Telangana

Abstract: *This project encompasses multiple integrated modules, each contributing to a secure and organized data management system involving three key stakeholders: the Administrator, Data Owner, and Data User. The admin module facilitates secure login using a user ID and password. Administrators can parse data content, manage user requests, and monitor user activities. All operations and activities are logged for security and auditing purposes. The Data Owner module enables users to register with their complete details and log in securely. Once authenticated, Data Owners can upload data files specifically in text format which are stored in a secured database. They can also generate and share encryption keys with authorized users and retain access to view their uploaded content. In the Data User module, users register and log in using an email ID and password. Data Users can search for available data uploaded by Data Owners. Upon receiving the correct decryption key, users can verify the key and proceed to decrypt and download the data. Security within this system is enhanced using the Advanced Encryption Standard (AES), a symmetric-key block cipher known for its robustness in encrypting and decrypting sensitive data. This encryption standard is widely adopted across applications such as secure transactions and cloud storage. To ensure data integrity and secure password storage, the system also incorporates hash algorithms, specifically SHA-256, which generates a 256-bit message digest, ensuring the integrity and authenticity of data. In managing and displaying data efficiently, the project implements sorting algorithms. These algorithms are essential for organizing data in a logical, accessible format either alphabetically or numerically. The choice of a suitable sorting algorithm depends on various criteria, including time complexity, space complexity, and algorithm stability, which are critical for optimizing performance in data analysis and database management systems. Together, these components establish a secure, efficient, and user-friendly data sharing and access system grounded in cryptographic principles and computational efficiency.*

Keywords: *Data Owner*

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. 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" 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. 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. 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

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 5, June 2025 Impact Factor: 7.67

interfaces for the other two types. Hadoop distributed file system (HDFS) 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.

II. LITERATURE REVIEW

B. W. Zhang, Z. Jin, H. P. Wang, et al (2023): 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.

A.-L. Peng, Y.-M. Tseng, and S.-S. Huang (2022): 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.

J. Bengtson, J. B. Jensen, F. Sieczkowski, et al (2021): The previous adversary models of public key cryptography usually have a nature assumption that permanent/temporary secret (private) keys must be kept safely and internal secret states are not leaked to an adversary. However, in practice, it is difficult to keep away from all possible kinds of leakage on these secret data due to a new kind of threat, called "side-channel attacks". By side-channel attacks, an adversary could obtain partial information of these secret data so that some existing adversary models could be insufficient. Indeed, the study of leakage-resilient cryptography resistant to side-channel attacks has received significant attention recently. Up to date, no work has been done on the design of leakage-resilient certificateless key encapsulation (LR-CL-KE) or public key encryption (LR-CL-PKE) schemes under the continual leakage model. In this article, we propose the first LR-CL-KE scheme under the continual leakage model. Moreover, in the generic bilinear group (GBG) model, we formally prove that the proposed LR-CL-KE scheme is semantically secure against chosen ciphertext attacks for both Type I and Type II adversaries.

Y. Tseng, J.-D. Wu, S.-S. Huang, and T.-T. Tsai (2023): FSCQ is the first file system with a machine-checkable proof (using the Coq proof assistant) that its implementation meets its specification and whose specification includes crashes. FSCQ provably avoids bugs that have plagued previous file systems, such as performing disk writes without sufficient barriers or forgetting to zero out directory blocks. If a crash happens at an inopportune time, these bugs can lead to data loss. FSCQ's theorems prove that, under any sequence of crashes followed by reboots, FSCQ will recover the file system correctly without losing data. To state FSCQ's theorems, this paper introduces the Crash Hoare logic (CHL), which extends traditional Hoare logic with a crash condition, a recovery procedure, and logical address spaces for

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, June 2025
Impact Factor: 7.67

specifying disk states at different abstraction levels. CHL also reduces the proof effort for developers through proof automation. Using CHL, we developed, specified, and proved the correctness of the FSCQ file system. Although FSCQ's design is relatively simple, experiments with FSCQ running as a user-level file system show that it is sufficient to run Unix applications with usable performance. FSCQ's specifications and proofs required significantly more work than the implementation, but the work was manageable even for a small team of a few researchers.

Z. Jin, B. W. Zhang, L. Zhang, et al (2023): 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.

Existing System

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. The CBS architecture consists of multiple layers, including the storage layer, block layer, and file system layer, which makes it difficult to model and reason about the system state.

Existing System Disadvantages:

- Poor integration with real systems.
- Inadequate tool support for block-level storage.
- High expertise requirement.
- Maintenance overhead.

Proposed System

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.

Proposed System Advantages:

- A fast and secure hash algorithm that produces a variable-length hash value.
- The output hash value is always the same for a given input.
- Better fault isolation.
- Enhances security verification.

III. SYSTEM ARCHITECTURE

The public cloud server has a login with a user id and password. It will validate the user id and password has a given correct. After successfully it will be login home page. It will display owner data information of all details. It has a Key

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, June 2025
Impact Factor: 7.67

Generator. The cloud server has the user information details. The cloud server has a request to the receiver key. It will have a Store all file information which has a stored in a Data Owner. Key Generator has a login with a user id and password. It will validate the user id and password has a given correct. After successfully it will be login home page. It will display a key test whether it is correct keys are not. It will verify keys. Owner data has a register with details and then login with a user id and password. Data Owner has to store files it will be stores at a database. User has a register with details and then login with a user id and password. Data User has search files and it will search from the database. Then we have a get a key response. The Data User has a key has a verification whether it has a correct keyword or not. Then the Data User has a decrypt a file download from the database.

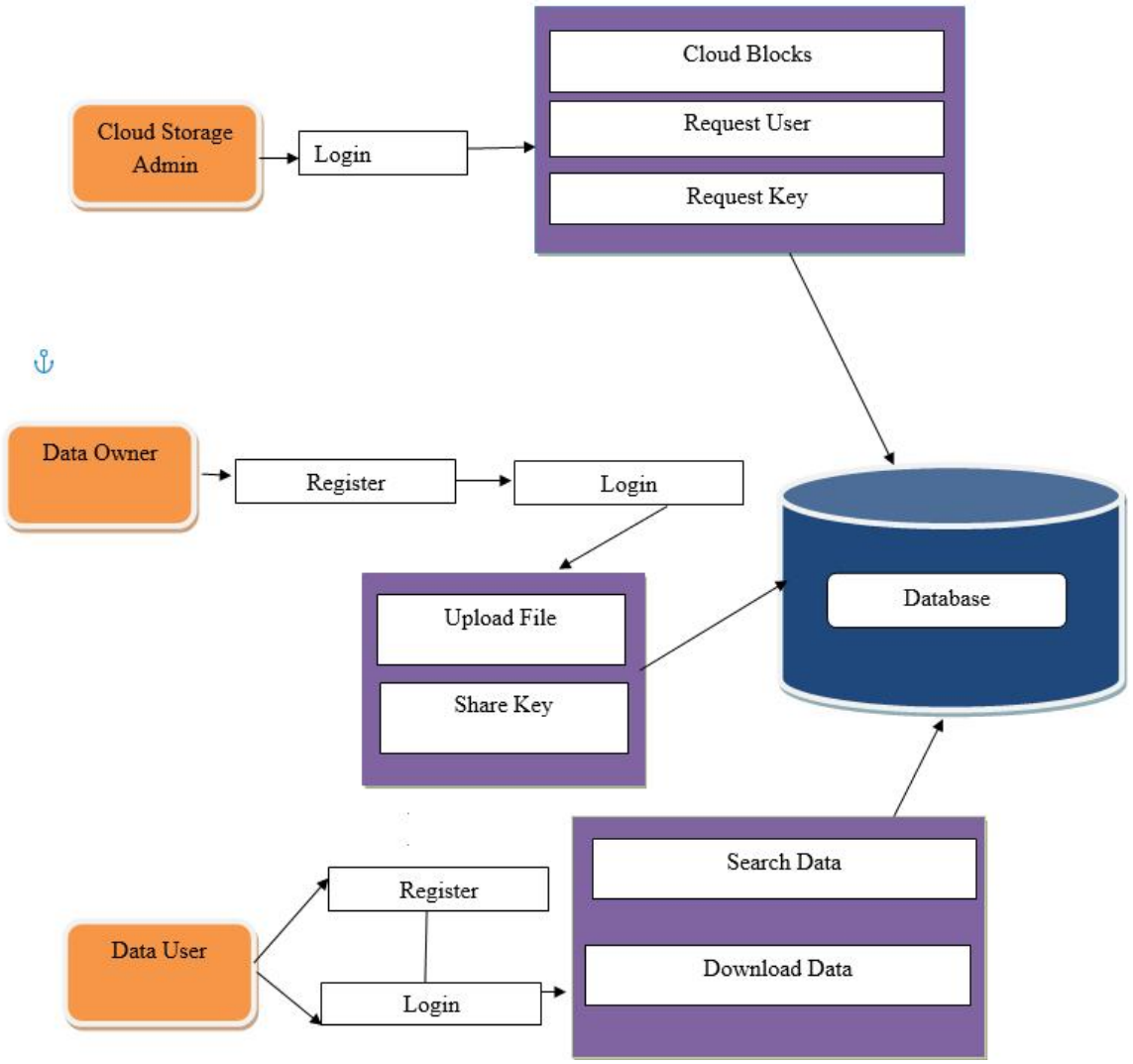


Figure: System Architecture

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, June 2025
Impact Factor: 7.67

IV. METHODOLOGY

MODULES:

1. User Interface:

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 second module of the project. 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 users. Admin can also have activity information and all recorded information.

3. Data owner:

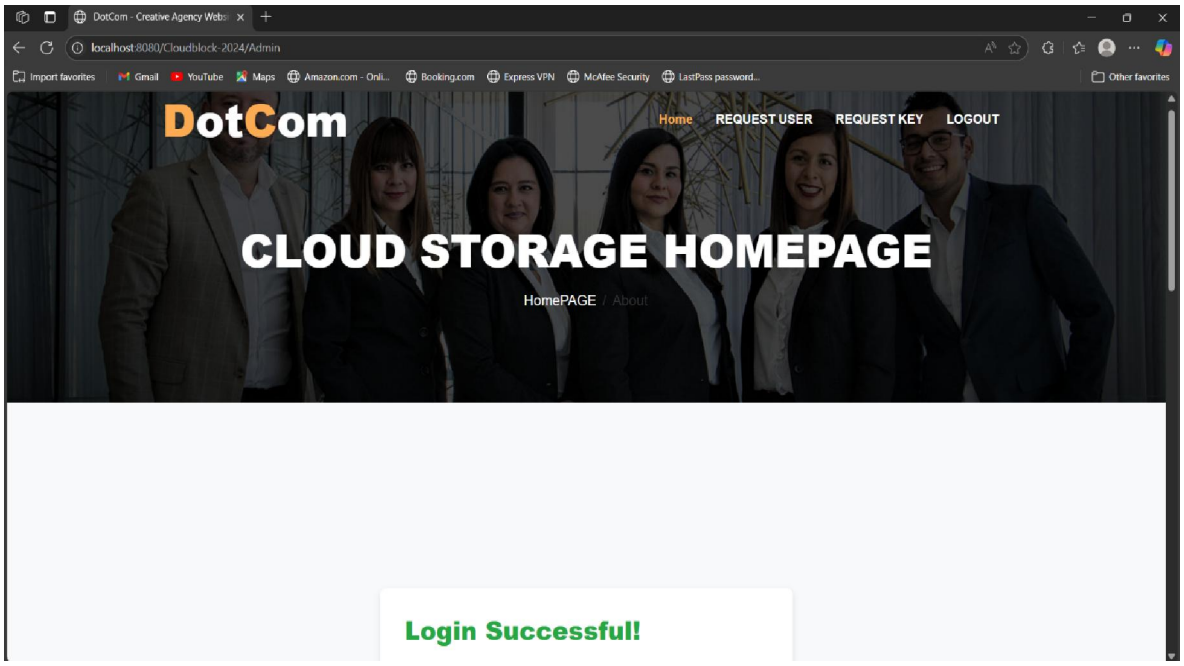
This is the third module of this project. In this project data owner has a register with all details and login with a user id and password. The Data owner has to upload a file the file has upload with a text file to the data. The data owner has a share a key to the user. The data owner has a view own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

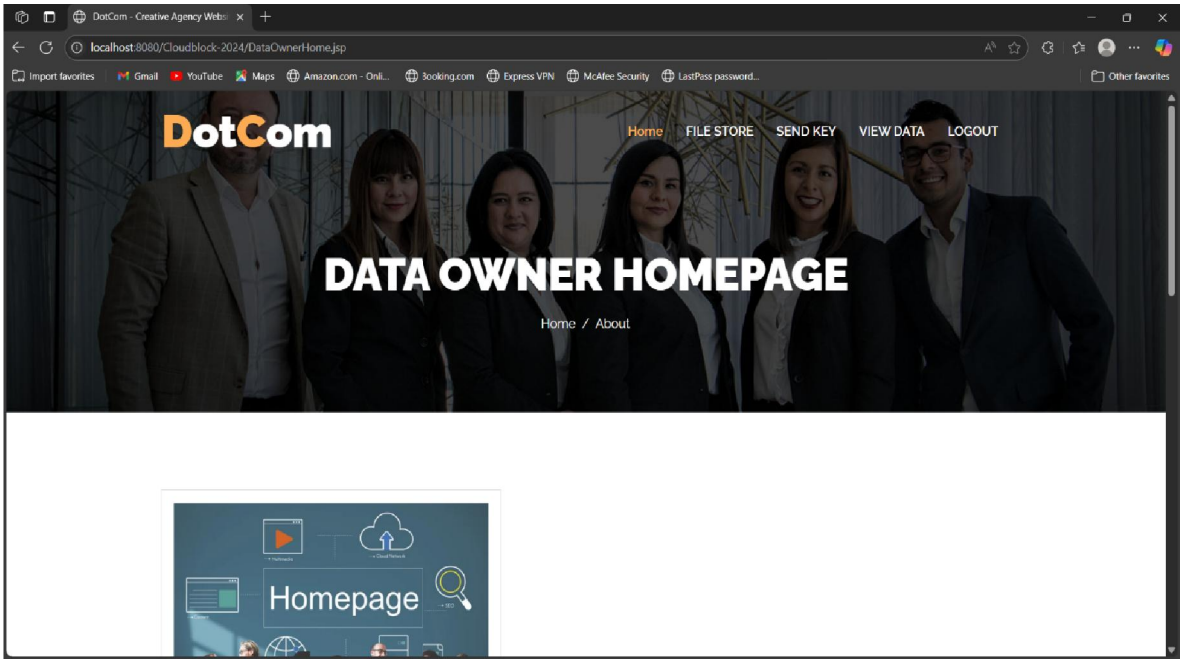
4. Data user:

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.

V. EXPERIMENTAL RESULT

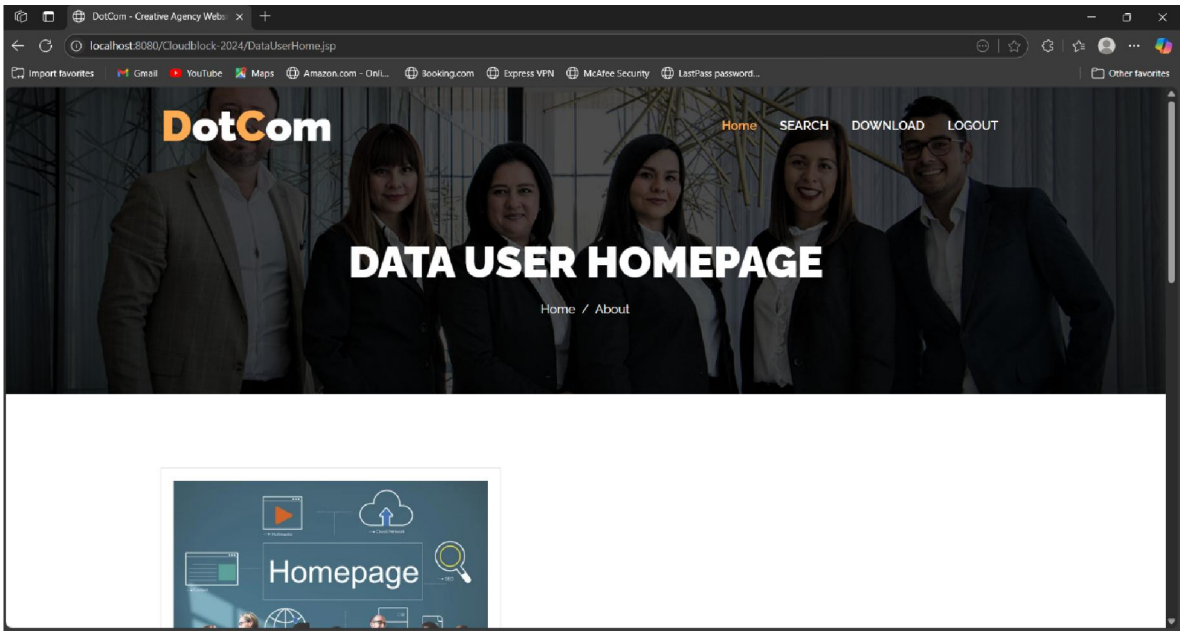
IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, June 2025
Impact Factor: 7.67





DOI: 10.48175/IJARSCT-27702

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 5, June 2025 Impact Factor: 7.67



VI. 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. Furthermore, the work lays a foundation for verifying more advanced properties such as crash consistency, concurrent correctness, and security guarantees, opening avenues for future research in formally verified cloud infrastructure. This contribution is a significant step toward developing dependable and mathematically verified storage systems in the era of large-scale cloud computing.

REFERENCES

[1] 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.
[2] M. Mesnier, G. R. Ganger, and E. Riedel, “Object-based storage,” IEEE Communications Magazine, vol. 41, no. 8, pp. 84–90, 2003.
[3] 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.
[4] 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.
[5] J. M. Wing, “A specifier’s introduction to formal methods,” Computer, vol. 23, no. 9, pp. 8–22, 1990.
[6] J. H. Gallier, Logic for Computer Science: Foundations of Automatic Theorem Proving, 2nd ed., Dover Publications, New York, NY, USA, pp. 1–12, 2015.
[7] J. C. Reynolds, “Separation logic: A logic for shared mutable data structures,” in Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science, Copenhagen, Denmark, pp. 55–74, 2002.
[8] P. O'Hearn, “Separation logic,” Communications of the ACM, vol. 62, no. 2, pp. 86–95, 2019.
[9] C. A. R. Hoare, “An axiomatic basis for computer programming,” Communications of the ACM, vol. 12, no. 10, pp. 576–580, 1969.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, June 2025
Impact Factor: 7.67

[10] S. C. Qin, Z. W. Xu, and Z. Ming, "Survey of research on program verification via separation logic," Journal of Software, vol. 28, no. 8, pp. 2010–2025, 2017. (in Chinese) D. Pym, J. M. Spring, and P. O'Hearn, "Why separation logic works," Philosophy & Technology, vol. 32, no. 3, pp. 483–516, 2019.
[11] N. A. Hamid and Z. Shao, "Interfacing Hoare logic and type systems for foundational proof-carrying code," in Proceedings of the 17th International Conference on Theorem Proving in Higher Order Logics, Park City, UT, USA, pp. 118–135, 2004.
[12] B. W. Zhang, Z. Jin, H. P. Wang, et al., "A tool for verifying cloud block storage based on separation logic," Journal of Software, vol. 33, no. 6, pp. 2264–2287, 2022. (in Chinese)
[13] J. Backfield, Becoming Functional: Steps for Transforming into A Functional Programmer. O'Reilly Media, Inc., Sebastopol, CA, USA, pp. 1–3, 2014.
[14] T. White, Hadoop: The Definitive Guide, 3rd ed., O'Reilly Media, Inc., Sebastopol, CA, USA, pp. 43–53, 2012.
[15] Apache, "HDFS truncate," Available at: https://issues.apache. org/jira/ browse/HDFS-3107, 2016.
[16] L. Cardelli and P. Wegner, "On understanding types, data abstraction, and polymorphism," ACM Computing Surveys, vol. 17, no. 4, pp. 471–523, 1985.
[17] B. Biering, L. Birkedal, and N. Torp-Smith, "BI hyper doctrines and higher-order separation logic," in Proceedings of the 14th European Symposium on Programming, Edinburgh, UK, pp. 233–247, 2005.
[18] C. Baier and J. P. Katoen, Principles of Model Checking. MIT Press, Cambridge, MA, USA, pp. 7–16, 2008.
[19] H. G. Chen, D. Ziegler, T. Chajed, et al., "Using Crash Hoare logic for certifying the FSCQ file system," in Proceedings of the 25th Symposium on Operating Systems Principles, Monterey, CA, USA, pp. 18–37, 2015.
[20] H. G. Chen, T. Chajed, A. Konradi, et al., "Verifying high performance crash-safe file system using a tree specification," in Proceedings of the 26th Symposium on Operating Sys
tems Principles, Shanghai, China, pp. 270–286, 2017.
[21] K. Arkoudas, K. Zee, V. Kuncak, et al., "Verifying a file system implementation," in Proceedings of the 6th International Conference on Formal Engineering Methods, Seattle, WA,
USA, pp. 373–390, 2004.
[22] P. Gardner, G. Ntzik, and A. Wright, "Local reasoning for the POSIX file system," in Proceedings of the 23rd European Symposium on Programming Languages and Systems, Grenoble, France, pp. 169–188, 2014.
[23] I. Pereverzeva, L. Laibinis, E. Troubitsyna, et al., "Formal modelling of resilient data storage in cloud," in Proceedings of the 15th International Conference on Formal Engineering Methods, Queenstown, New Zealand, pp. 363–379, 2013.
[24] R. Bobba, J. Grov, I. Gupta, et al., "Survivability: Design, formal modeling, and validation of cloud storage systems using Maude," in Assured Cloud Computing, R. H. Campbell, C. A. Kamhoua, and K. A. Kwiat, Eds. Wiley-IEEE Press, Hoboken, NJ, USA, pp. 10–48, 2018