SECURITY CLOUD STORAGE BASED ON RLWE (RING LEARNING WITH ERRORS)

[1]Namana.Murali Krishna, [2]Sujeeth T
[1]Professor, Dept of CSE, Vignan Institute of Technology and Science, Hyderabad – 508284.
[2]Assistant Professor, Dept of CSE, Siddhartha Educational Academy Group of Institutions, Tirupati - 517501.
[1]muralinamana@gmail.com , [2]sujeeth.2304@gmail.com

ABSTRACT

With the rapid development of cloud storage and quantum computing, ensuring the integrity of outsourced data for data owners becomes a serious concern. To address this problem, existing protocols for auditing cloud storage are usually based on post-quantum cryptographies to check data integrity. Nevertheless, these protocols employ heavy cryptographic operations to construct the data tags, making their efficiency low and extensibility poor. In this paper, we take a new perspective and explore the possibility of designing secure cloud storage (SCS) protocols based on the ring learning with errors (RLWE) problem. Instead of matrix variables, our protocol only utilizes vector variables to generate data block tags so that it has a much lower computational complexity. We then give a strict security proof of cheating resistance against the malicious cloud and privacy guarantee against the curious third-party auditor. We also extend the proposed protocol to support data dynamics and batch auditing for more application scenarios. As a further contribution, we summarize a systematic framework for designing lattice-based SCS protocols. Finally, exhaustive performance analysis and comparison are provided to verify that the proposed protocol outperforms the existing lattice-based SCS protocols in terms of both operational efficiency and functional extensibility.

1. INTRODUCTION

In our paper Cloud storage is becoming a common practice thanks to the rising popularity of cloud computing [1]. However, outsourcing data to the cloud also brings new security challenges. Firstly, despite strong security measures taken by cloud services providers (CSP), outsourced data still faces a large amount of internal and external attacks [2] [3]. Secondly, CSPs are motivated to discard data that are rarely accessed to reclaim storage for more profit [4] [5]. Thirdly, in case of data loss incidents, CSPs have incentives to hide the truth and cheat that the outsourced data are still undamaged for their own benefits [6] [7]. Therefore, it is important for data owners to have the ability to check the integrity of the outsourced data. A secure cloud storage (SCS) protocol should satisfy several substantial requirements [8]– [11]. 1) Public auditing. An SCS protocol should support public auditing. Specifically, the integrity auditing task can be outsourced to a third-party auditor (TPA), which is more professional and fairer in case of data loss. 2) High Efficiency. The operating efficiency of an SCS protocol should be as high as possible. 3) Strong Extensibility. An SCS protocol should be easily extended to support other common functions, such as data dynamics and batch auditing. In summary, it is better for an SCS protocol to involve only light cryptographic operations, and support public auditing, data dynamics, and batch auditing. In addition, an SCS protocol also requires that data owners have the ability to verify data integrity without physically possessing the actual data, which can hugely relieve the userside storage burden.

Previous SCS protocols always take advantage of classic cryptographies, such as RSA and discrete logarithm problem. Specifically, these protocols explore classic cryptographies to generate the outsourced data and the integrity proof, which are utilized to perform the data auditing task. However, these SCS protocols can no longer work well in front of quantum attacks. To address this problem, the research direction is shifting to design SCS protocols based on post-quantum cryptographies. As we know, lattice cryptography is one of the most famous post-quantum cryptographies, which is widely accepted to be secure against quantum attacks [12] [13]. Thus, the anti-quantum computing SCS protocols are usually designed based on lattice cryptographies [14]- [17]. Nevertheless, all of the existing lattice-based protocols compute data tags using matrix variables, which results in low efficiency and poor extensibility. In stark contrast, in this paper, we propose to design the SCS protocol based on equality checking in the ring learning with error (RLWE) problem, utilizing only vector variables to generate data tags. Our design has both conceptual and technical novelty. We first propose an SCS protocol based on RLWE. Then, we present two security games to formalize a security model of SCS protocols. These two games are designed based on cheating resistance against the malicious cloud and privacy guanrantee against the curious TPA respectively, which captures the security

intuition in practice. Then, we propose an efficient algorithm to support data dynamics, which includes inserting, deleting and modifying user data. Specifically, we utilize a constant-size cache to maintain the relationship between the indices of data blocks and their tags. In this way, tag re-computation after each data update can be avoided and then data dynamics can be handled efficiently. Through aggregating multiple verification equations into a single one, the proposed SCS protocol can efficiently support batch auditing, i.e., the TPA can audit multiple users' data at the same time. Finaly, we establish a generic framework for designing SCS protocols based on lattice cryptographies by exploring the intrinsic relationship between SCS and lattice cryptographies.

2. SYSTEM ANALYSIS

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of software development methodologies.

2.1 Existing System

In fact, the data deduplication technique, which is widely adopted by current cloud storage services in existing clouds, is one example of exploiting the similarities among different data chunks to save disk space and avoid data retransmission . It identifies the same data chunks by their fingerprints which are generated by fingerprinting algorithms such as SHA-1, MD5. Any change to the data will produce a very different fingerprint with high probability . However, these fingerprints can only detect whether or not the data nodes are duplicate, which is only good for exact equality testing. Determining identical chunks is relatively straightforward but efficiently determining similarity between chunks is an intricate task due to the lack of similarity preserving fingerprints (or signatures).

a) Disadvantages of Existing System

- Unplanned distribution of data chunks can lead to high information disclosure even while using multiple clouds.
- Frequent modifications of files by users result in large amount of similar chunks1;
- Similar chunks across files, due to which existing CSPs use the data de duplication technique.

2.2 Proposed System

We present Store Sim, an information leakage aware multi cloud storage system which incorporates three important distributed entities and we also formulate information leakage optimization problem in multi cloud. We propose an approximate algorithm, BFS Minhash, based on Minhash to generate similarity-preserving signatures for data chunks. Based on the information match measured by BFS Minhash, we develop an efficient storage plan generation algorithm, clustering, for distributing users data to different clouds.

a) Advantages

- However, previous works employed only a single cloud which has both compute and storage capacity. Our work is different since we consider a mutli cloud in which each storage cloud is only served as storage without the ability to compute.
- Our work is not alone in storing data with the adoption of multiple CSPs these work focused on different issues such as cost optimization , data consistency and availability.

2.3 Architecture analysis:

Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a

new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

2.4 Data Preprocessing

There are three symbolic data types in NSL-KDD data features: protocol type, flag and service. We use one-hot encoder mapping these features into binary vectors. One-Hot Processing: NSL-KDD dataset is processed by one-hot method to transform symbolic features into numerical features. For example, the second feature of the NSL-KDD data sample is protocol type. The protocol type has three values: tcp, udp, and icmp. One-hot method is processed into a binary code that can be recognized by a computer, where tcp is [1, 0, 0], udp is [0, 1, 0], and icmp is [0, 0, 1]

3. SYSTEM DESIGN

3.1 System architecture

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

Figure-1: Architecture diagram

a) 3-Tier Architecture:

The three-tier software architecture (a three layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging. The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

b) Construction of Use case diagrams

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

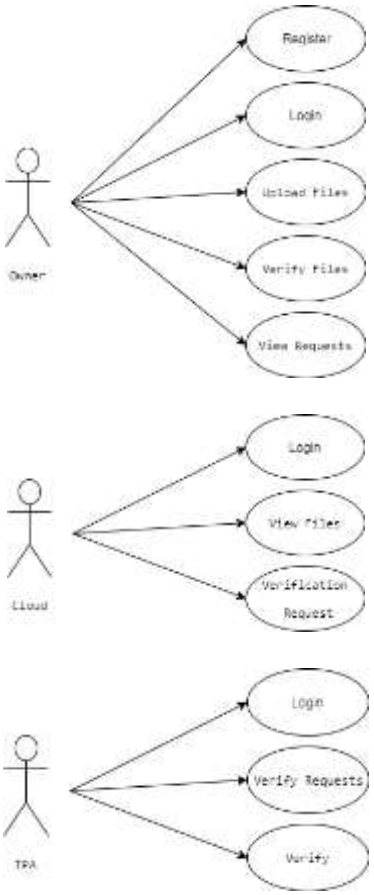

Figure-2: Use Case Diagram

c) Sequence Diagrams

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram

that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

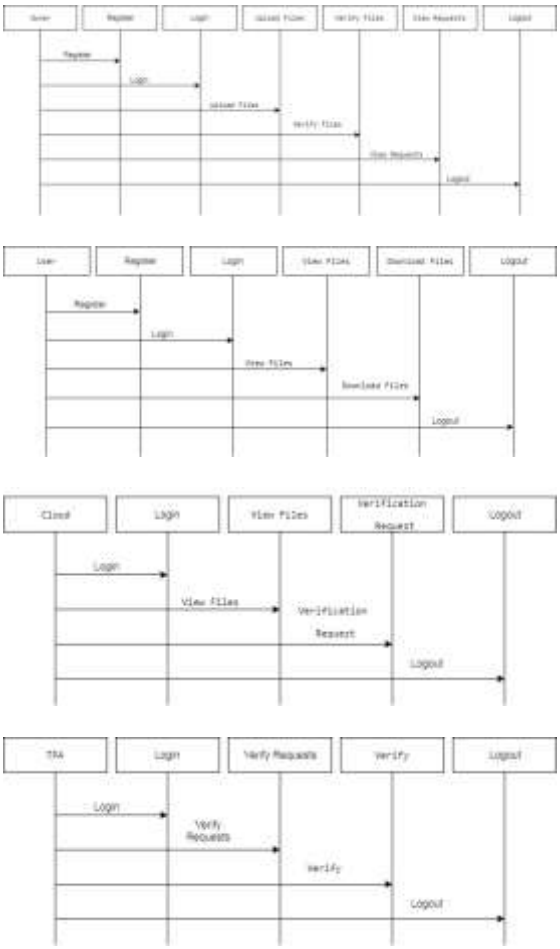

Figure-3: Sequence diagram

d) Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

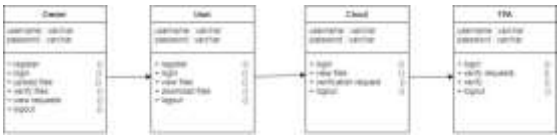

Figure-4: Class Diagram

e) Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

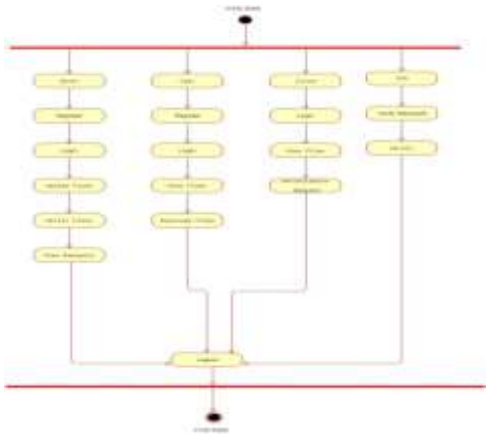

Figure-5: Activity Diagram

4. IMPLEMENTATION

a) Design

The software system design is produced from the results of the requirements phase. Architects have the ball in their court during this phase and this is the phase in which their focus lies. This is where the details on how the system will work is produced. Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

b) Implementation

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle. For a developer, this is the main focus of the life cycle because this is where the code is produced. Implementation my overlap with both the design and testing phases. Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

5. TESTING

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. The following is the description of the

testing strategies, which were carried out during the testing period.

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. Unit tests and system/acceptance tests are done during this phase. Unit tests act on a specific component of the system, while system tests act on the system as a whole.

So in a nutshell, that is a very basic overview of the general software development life cycle model. Now let's delve into some of the traditional and widely used variations.

5.1 System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

5.2 Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

5.3 Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

5.4 Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

6. OUTPUT SCREENS

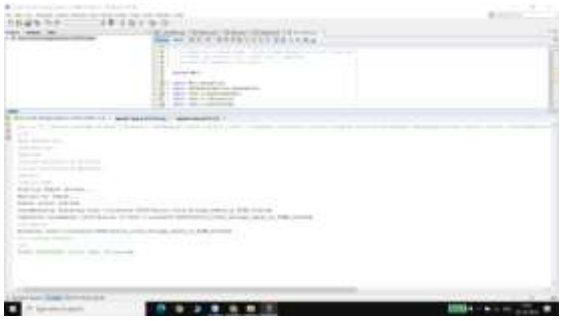

Figure: Home Page

7. CONCLUSION

Distributing data on multiple clouds provides users with a certain degree of information leakage control in that no single cloud provider is privy to all the user's data. However, unplanned distribution of data chunks can lead to avoidable information leakage. We show that distributing data chunks in a round robin way can leak user's data as high as 80% of the total information with the increase in the number of data synchronization. To optimize the information leakage, we presented the StoreSim, an information leakage aware storage system in the multicloud. StoreSim achieves this goal by using novel algorithms, BFSMinHash and SPClustering, which place the data with minimal information leakage (based on similarity) on the same cloud. Through an extensive evaluation based on two real datasets, we demonstrate that StoreSim is both effective and efficient (in terms of time and storage space) in minimizing information leakage

during the process of synchronization in multicloud. We show that our StoreSim can achieve near-optimal performance and reduce information leakage up to 60% compared to unplanned placement. Finally, through our attackability analysis, we further demonstrate that StoreSim not only reduces the risk of wholesale information leakage but also makes attacks on retail information much more complex.

REFERENCES

[1] K. Akher_, M. Gerndt, and H. Harroud, ``Mobile cloud computing for computation of_oading: Issues and challenges," *Appl. Comput. Informat.*,vol. 14, no. 1, pp. 1_16, 2018.

[2] E. Ahmed, A. Gani, M. K. Khan, R. Buyya, and S. U. Khan, ``Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges," *J. Netw. Comput. Appl.*, vol. 52, pp. 154_172, Jun. 2015.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, ``Mobile edge computing_A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1_16, 2015.

[4] R. Roman, J. Lopez, and M. Mambo, ``Mobile edge computing, Fog *et al.*: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680_698, Jan. 2018.

[5] R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, and G. Mastorakis, ``Drop computing: Ad-hoc dynamic collaborative computing," *Future Gener. Comput. Syst.*, vol. 92, pp. 889_899, Mar. 2017.

[6] V.-C. Tabusca, R.-I. Ciobanu, and C. Dobre, ``Data consistency in mobile collaborative networks based on the drop computing paradigm," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE)*, Oct. 2018, pp. 29_35.

[7] G. Huerta-Canepa and D. Lee, ``A virtual cloud computing provider for mobile devices," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Services Social Netw. Beyond (MCS)*. New York, NY, USA: ACM, 2010, pp. 6:1_6:5. doi: 10.1145/1810931.1810937.

[8] N. Fernando, S. W. Loke, and W. Rahayu, ``Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput. (UCC)*,Washington, DC, USA: IEEE Comput. Soc.,Dec. 2011, pp. 281_286. doi: 10.1109/UCC.2011.45.

[9] E. Miluzzo and R. Cáceres, and Y.-F. Chen, ``Vision: mClouds_Computing on clouds of mobile devices," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services (MCS)*. New York, NY,USA: ACM, 2012, pp. 9_14. doi: 10.1145/2307849.2307854.