ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

Integrity Verification of Cooperative Provable Data Using Cloud Storage Factors

Podila Brijesh Babu
Nimra College of Engineering & Technology.

Padmaja
Nimra College of Engineering & Technology.

ABSTRACT:

Provable data possession (PDP) is a technique for ensuring the integrity of data in storage outsourcing. In this paper, we address the construction of an efficient PDP scheme for distributed cloud storage to support the scalability of service and data migration, in which we consider the existence of multiple cloud service providers to cooperatively store and maintain the clients' data. We present a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy.

We prove the security of our scheme based on multi-prover zero-knowledge proof system, which can satisfy completeness, knowledge soundness, and zero-knowledge properties. In addition, we articulate performance optimization mechanisms for our scheme, and in particular present an efficient method for selecting optimal parameter values to minimize the computation costs of clients and storage service providers. Our experiments show that our solution introduces lower computation and communication overheads in comparison with non-cooperative approaches.

1. INTRODUCTION:

There exist various tools and technologies for multi cloud, such as Platform VM Orchestrator, VMwarevSphere, and Ovirt. These tools help cloud providers construct a distributed cloud storage platform for managing clients' data. However, if such an important platform is vulnerable to security attacks, it would bring irretrievable losses to the clients. For example, the confidential data in an enterprise may be illegally accessed through a remote interface provided by a multi-cloud, or relevant data and archives may be lost or tampered with when they are stored into an· uncertain storage pool outside the enterprise. Therefore, it is indispensable for cloud service providers to provide security techniques for managing their storage services.

2. SYSTEM DESIGN:

This section would analyze the system based on the requirement gathered in the previous section. Analysis would concentrate on what the system should have and should do rather than how it is going to be done. Basically, there are two areas of analysis: data analysis and data flow analysis.

2.1 SOFTWARE TECHNIQUES:

A) OBJECT ORIENTED ANALYSIS AND DESIGN:

Object-oriented analysis and design (OAD) is often part of the development of large scale systems and programs often using the Unified Modeling Language (UML). OAD applies object-modeling techniques to analyze the requirements for a context for example, a system, and a set of system modules, an organization, or a business unit and to design a solution. Most modern object-oriented analysis and design methodologies are use case driven across requirements, design, implementation, testing, and deployment.Use cases were invented with object oriented programming, but they're also very well suited for systems that will be implemented in the procedural paradigm. The Unified Modeling Language (UML) has become the standard modeling language used in object-oriented analysis and design to graphically illustrate system concepts. Part of the reason for OAD is its use in developing programs that will have an extended lifetime.

B) Object Oriented Systems:

An object-oriented system is composed of objects. The behavior of the system is achieved through collaboration between these objects, and the state of the system is the combined state of all the objects in it. Collaboration between objects involves those sending messages to each other. The exact semantics of message sending between objects varies depending on what kind of system is being modeled.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

In some systems, "sending a message" is the same as "invoking a method". In other systems, "sending a message" might involve sending data via a socket.

C) Object Oriented Analysis:

Object-Oriented Analysis (OOA) aims to model the problem domain, the problem we want to solve by developing an object-oriented (OO) system. The source of the analysis is a written requirement statements, and/or written use cases, UML diagrams can be used to illustrate the statements. An analysis model will not take into account implementation constraints, such as concurrency, distribution, persistence, or inheritance, nor how the system will be built.

The model of a system can be divided into multiple domains each of which are separately analyzed, and represent separate business, technological, or conceptual areas of interest. The result of object-oriented analysis is a description of what is to be built, using concepts and relationships between concepts, often expressed as a conceptual model. Any other documentation that is needed to describe what is to be built is also included in the result of the analysis. That can include a detailed user interface mock-up document. The implementation constraints are decided during the object-oriented design (OOD) process.

D) Object Oriented Design:

Object-Oriented Design (OOD) is an activity where the designers are looking for logical solutions to solve a problem, using objects. Object-oriented design takes the conceptual model that is the result of object-oriented analysis, and adds implementation constraints imposed by the environment, the programming language and the chosen tools, as well as architectural assumptions chosen as basis of design. The concepts in the conceptual model are mapped to concrete classes, to abstract interfaces in APIs and to roles that the objects take in various situations. The interfaces and their implementations for stable concepts can be made available as reusable services. Concepts identified as unstable in object-oriented analysis will form basis for policy classes that make decisions, implement environment-specific or situation specific logic or algorithms. The result of the object-oriented design is a detail description how the system can be built, using objects.

Object-oriented software engineering (OOSE) is an object modeling language and methodology. OOSE was developed by Ivar Jacobson in 1992 while at Objectory AB After Rational bought Objectory AB, the OOSE notation, methodology, and tools became superseded.

- As one of the primary sources of the Unified Modeling Language (UML), concepts and notation from OOSE have been incorporated into UML.

- The methodology part of OOSE has since evolved into the Rational Unified Process (RUP).

- The OOSE tools have been replaced by tools supporting UML and RUP.

OOSE has been largely replaced by the UML notation and by the RUP methodology.

3) MODULE DESCRIPTION:

A) MULTI CLOUD STORAGE:

Distributed computing is used to refer to any large collaboration in which many individual personal computer owners allow some of their computer's processing time to be put at the service of a large problem. In our system the each cloud admin consist of data blocks. The cloud user uploads the data into multi cloud.

Cloud computing environment is constructed based on open architectures and interfaces; it has the capability to incorporate multiple internal and/or external cloud services together to provide high interoperability. We call such a distributed cloud environment as a multi-Cloud .A multi-cloud allows clients to easily access his/her resources remotely through interfaces.

B) COOPERATIVE PDP:

Cooperative PDP (CPDP) schemes adopting zero-knowledge property and three-layered index hierarchy, respectively. In particular efficient method for selecting the optimal number of sectors in each block to minimize the computation costs of clients and storage service providers. Cooperative PDP (CPDP) scheme without compromising data privacy based on modern cryptographic techniques

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

C) DATA INTEGRITY:

Data Integrity is very important in database operations in particular and Data warehousing and Business intelligence in general. Because Data Integrity ensured that data is of high quality, correct, consistent and accessible.

D) THIRD PARTY AUDITOR:

Trusted Third Party (TTP) whois trusted to store verification parameters and offerpublic query services for these parameters. In our system the Trusted Third Party, view the user data blocks and uploaded to the distributed cloud. In distributed cloud environment each cloud has user data blocks. If any odification tried by cloud owner a alert is send to the Trused Third Party.

E) CLOUD USER:

The Cloud User who has a large amount of data to be stored in multiple clouds and have the permissions to access and manipulate stored data. The User's Data is converted into data blocks. The data blocks are uploaded to the cloud. The TPA views the data blocks and Uploaded in multi cloud. The user can update the uploaded data. If the user wants to download their files, the data's in multi cloud is integrated and downloaded.

4. DATA-FLOW DIAGRAM (DFDS):

Data flow diagrams ("bubble charts") are directed graphs in which the nodes specify processing activities and the arcs specify data items transmitted between processing nodes. A data flow diagram might represent data flow between individual statements or blocks of statements in a routine, data flow between sequential routines, data flow between concurrent process, or data flow in a distributed computing system, where each node represents a geographically remote processing unit.

Unlike flow charts, data flow diagrams do not indicate decision logic or condition under which various nodes in the diagram might be activated. Data flow diagrams can be expressed using special symbols. These symbols can be used to denote processing nodes, data nodes, and data sources and data stores

Context Level DFD (0-Level)

Detailed level DFD (Level-1)

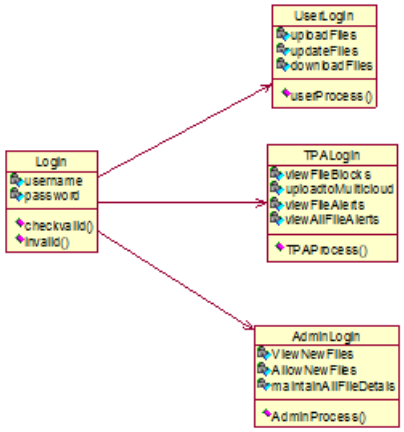
5. UNIFIED MODELING LANGUAGE:

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system. The UML is a appropriate for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real time embedded systems.

It is a very expressive language, addressing all the views needed to develop and then deploy such systems. The UML is only a language and so is just one part of a software development method. The UML is process independent, although optimally it should be used in a process that is use case driven, architecture-centric, iterative, and incremental.

Class Diagram:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal



Database Relations:

Oracle 7.3 is a relational database and data driven not design driven. Designed once a relational database the data changes over time does not affect the applications. In oracle 7.3 data is self-describing.

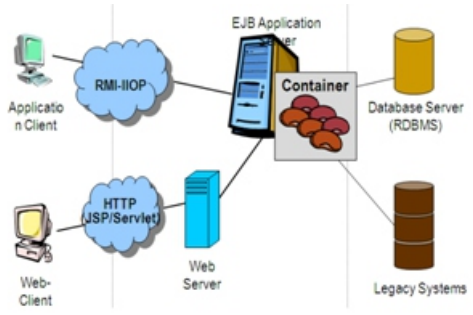
Data stored in one place read from one place and modified in one place. Data is stored once so maintaining consistency. The objective of oracle 7.3 is to manage corporate data, no matter what the type of data type it is. This includes structured data and unstructured data.

Oracle focuses mainly on the following areas

1.High-end online-transaction processing and data warehousing requirements.

2.Object-relational extensions.

3.Performance, manageability, and functional enhancements.



Steps in the execution of a JSP Application:

1.The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.

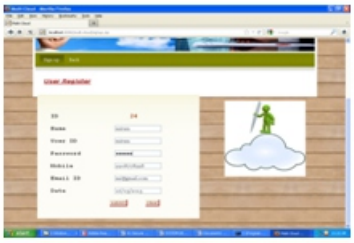
2.This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.

3. JSP engine is program which can under stands the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

TEST RESULTS:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal



CONCLUSION:

We presented the construction of an efficient PDP scheme for distributed cloud storage. Based on homomorphic verifiable response and hash index hierarchy, we have proposed a cooperative PDP scheme to support dynamic scalability on multiple storage servers.

We also showed that our scheme provided all security properties required by zero knowledge interactive proof system, so that it can resist various attacks even if it is deployed as a public audit service in clouds. Furthermore, we optimized the probabilistic query and periodic verification to improve the audit performance.

Our experiments clearly demonstrated that our approaches only introduce a small amount of computation and communication overheads. Therefore, our solution can be treated as a new candidate for data integrity verification in outsourcing data storage systems.

As part of future work, we would extend our work to explore more effective CPDP constructions. Finally, it is still a challenging problem for the generation of tags with the length irrelevant to the size of data blocks. We would explore such a issue to provide the support of variable-length block verification.

REFERENCES:

[1] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. T. Foster, Virtual infrastructure management in private and hybrid clouds," IEEE Internet Computing, vol. 13, no. 5, pp. 14–22,2009.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

[2] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in ACM Conference on Computer and Communications Security, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 598–609.

[3] A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in ACMConference on Computer and Communications Security, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds. ACM, 2007, pp. 584–597.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalabl and efficient provable data possession," in Proceedings of the 4th international conference on Security and privacy in communication netowrks, SecureComm, 2008.

[5] C. C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in ACM Conference on Computer and Communications Security, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 213–222.

[6] H. Shacham and B. Waters, "Compact proofs of retrievability," in ASIACRYPT, ser. Lecture Notes in Computer Science, J. Pieprzyk, Ed., vol. 5350. Springer, 2008, pp. 90–107.

[7] Q. Wang, C.Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in

[8] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in SAC, W. C. Chu, W. E. Wong, M. J. Palakal, and C.-C. Hung, Eds. ACM, 2011, pp. 1550–1557.

[9] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in ACM Conference on Computer and Communications Security, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. ACM, 2009, pp. 187–198.

[10] Y. Dodis, S. P. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in TCC, ser. Lecture Notes in Computer Science, O. Reingold, Ed., vol. 5444. Springer, 2009, pp. 109–127.

[11] L. Fortnow, J. Rompel, and M. Sipser, "On the power of multiprover interactive protocols," in Theoretical Computer Science, 1988, pp. 156–161.

[12] Y. Zhu, H. Hu, G.-J. Ahn, Y. Han, and S. Chen, "Collaborative integrity verification in hybrid clouds," in IEEE Conference on the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, Orlando, Florida, USA, October 15-18, 2011, pp. 197–206.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep., Feb 2009