ENCRYPTED DATA WITH OWNERSHIP MANAGEMENT IN CLOUD

[1]M.Mahima, [2]DR.C.CHELLAPPAN

[1]PG Scholar, [2]Principal
Department of Computer Science and Engineering,
GKM Engineering College, Chennai

***ABSTRACT*: Data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data, and has been widely used in cloud storage to reduce the amount of storage space and save bandwidth. To protect the confidentiality of sensitive data while supporting de duplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. To better protect data security, the first attempt to formally address the problem of authorized data de duplication. Different from traditional de duplication systems, the differential privileges of users are further considered induplicate check besides the data itself. We also present several new de duplication constructions supporting authorized duplicate check in hybrid cloud architecture. Security analysis demonstrates that our scheme is secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments using our prototype. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.**

***Keywords*: Deduplication, BIG DATA, cloud.**

I. INTRODUCTION

Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Every IT decision maker either wants deduplication (dedupe) or needs it. At least that is what they are being told by the market and vendors who are trying to sell new functionality. The fact is, while deduplication can save backend storage, it is not a fit for everyone. Let's dive into the subject of dedupe and figure out if it is right for you.

1.1 DATA DEDUPLICATION TYPES:

Data deduplication has five types.
(i) Compression, (ii) Single-instance storage, (iii) Sub-file deduplication, (iv) File level deduplication, (v) Block level deduplication

1.1.1 Data Compression

Data compression does not recognize or eliminate duplicate file it just compress the given file by reducing the size of files. Data comp1ression works within a file to identify and remove empty space that appears as repetitive patterns. That deduplication is local to the file and remains independent of other files and data segments within those files. Benefits of Data compression are limited though it has been available for many years; it becomes isolated to each particular file. For example, data compression cannot identify and remove duplicate files, but will independently compress each of the files.

1.1.2 Single-Instance Storage

Single-Instance Storage removes multiple copies of any file. Single-instance storage (SIS) environments can detect and eliminate redundant copies of identical files. As name suggests it keeps only single Instance or copy of data and pointers are created for all other users who own the same file. In Single-instance storage systems, content of files are checked to determine if the file to be uploaded on cloud is identical to an existing file or not. The number of files that are stored as unique at cloud, on the basis of file content in Single Instance system, there may be large amount of redundancy in that file or files. For example, a new date inserted into the title slide of a presentation of some files, this is very small amount of change in huge files but considered as different files to be stored without further de-duplication.

1.1.3 Sub-file Deduplication

If redundant data exists in separate files not needed to be identical files, that redundancy can be avoided with the help of Sub-file Deduplication. Sub-file deduplication detects redundant data within and across files which is not the case in SIS implementations. Using sub-file deduplication, redundant copies of data are found and are removed even after the duplicated data exist, within non identical files. As a result, sub-file deduplication eliminates the storage of duplicate data across an organization. Though files are not identical, Sub-file data de-duplication has large amount of benefits, have data elements that are already recognized somewhere in the organization. Sub-file deduplication implementation has two types. (i) Fixed length sub-file deduplication, (ii) Variable length implementations

Fixed length sub-file deduplication uses fixed length of data to search for the duplicate data within the files. Fixed length segments are simple in design, but miss many opportunities to discover redundant sub-file data. For example, when name of person is added to certain file's tile page the whole content of the file will shift, resulting the failure of the de-duplication to detect equivalencies. Means, small change or addition in file may cause non equivalencies.

Variable length implementations are usually not corresponding to segment length. Variable length implementations match data segment sizes to the naturally occurring duplication within files, vastly increasing the overall de-duplication ratio (In the example above, variable-length deduplication will catch all duplicate segments in the document, no matter where the changes occur). So data deduplication is widely used in most of the organizations, which is also called as, data reduction, single instance storage, capacity optimized storage and intelligent compression and removes all but keeps one copy which creates pointers to the file so that users could access the file as and when needed.

1.1.4. File-level deduplication

File level deduplication checks certain attributes of given files against the specific index, if the file is unique, updating are done in It is commonly known as single-instance storage, if file is not unique then only a pointer corresponding to existing file that is stored references is updated. Only the single instance of file is saved instead of saving duplicate copies of files.

1.1.5. Block-level deduplication

In Block-level data deduplication, file is being divided into segments blocks or chunks, those chunks of data are checked for redundancies or previously stored information. The size of the chunk which needs to be checked varies from vendor to vendor. Each chunk is assigned with an identifier, by using hash algorithm for example – it generates a unique ID to that particular block. The comparison takes place between that identifier and particular unique Id. If ID is already present, it indicates that data is processed before. Therefore only a pointer reference is saved to the previously stored data. If the ID is new and does not exist, then that block is unique. The unique chunk is stored and the unique ID is updated in the Index. Some will have fixed block sizes, while some others use variable block sizes likewise few may also change the size of fixed block size for sake of confusing. Block sizes of fixed size may vary from 8KB to 64KB but the main difference with it is the smaller the chunk, than it will be likely to have opportunity to identify it as the duplicate data. If less data is stored than it obviously means greater reductions in the data that is stored. The only major issue by using fixed size blocks is that in case if the file is modified and the de-duplication result uses the same previously inspected result than there will be chance of not identifying the same redundant data segment, as the blocks in the file would be moved or changed, than they will shift downstream from change, by offsetting the rest of comparisons.

II. METHODOLOGY:

There are four major steps involved in data deduplication. The chunking method splits the incoming large data into small portions or 'chunks'. Using hash algorithm, a unique identifier value is assigned to each chunk. The new incoming chunks are compared with the existing stored chunk using the unique identfier. If the identfier matches, the redundant chunk will be removed and will be given the reference point; otherwise, new chunk will be stored. The key objective of the chunking algorithm is to divide the data object into small fragments. The data object may be a file, a data stream, or some other form of data. There are different chunking algorithms for deduplication including file-level chunking, block-level chunking, content-based chunking, sliding window chunking, and TTTD chunking.

2.1 Chunking Methods

Efficient chunking is one of the key elements that decide the overall deduplication performance. There are a number of methodologies to detect duplicate chunk of data using fixed-level chunking and fixed-level chunking using rolling check-sums.

2.1.1 File-Level Chunking

File-level chunking or whole file chunking considers an entire file as a chunk, rather than breaking files into multiple chunks. In this method, only one index is created for the complete file and the same is compared with the already stored whole file indexes. As it creates one index for the whole file, this approach stores less number of index values, which in turn saves space and helps store more index values compared to other approaches. It avoids maximum metadata lookup overhead and CPU usage. Also, it reduces the index lookup process as well as the I/O operation for each chunk. However, this method fails when a small portion of the file is changed. Instead of computing the index for the changed parts, it calculates the index for the entire file and moves it to the backup location. Hence, it affects the throughput of the deduplication system. Especially for backup systems and large files that change regularly, this approach is not suitable.

2.1.2 Block Level Chunking

Fixed-Size Chunking: Fixed-size chunking method splits files into equally sized chunks. The chunk boundaries are based on offsets like 4, 8, 16 KB, etc. These methods effectively solve issues of the file-level chunking method. If a huge file is altered in only a few bytes, only the changed chunks must be reindexed and moved to the backup location. However, this method creates more chunks for larger file which requires extra space to store the metadata and the time for lookup of metadata is more. As it splits the file into fixed size, byte shifting problem occurs for the altered file. If the bytes are inserted or deleted on the file, it changes all subsequent chunk position which results in duplicate index values. This can be eliminated by using bit-by-bit comparison which is more accurate, but requires more time to compare the files.

Variable Size Chunking: The files can be broken into multiple chunks of variable sizes by breaking them up based on the content rather than on the fixed size of the files. This method resolves the fixed chunk size issue. When working on a fixed chunking algorithm, fixed boundaries are defined on the data based on chunk size which do not alter even when the data are changed. However, in the case of a variable-size algorithm different boundaries are defined, which are based on multiple parameters that can shift when the content is changed or deleted. Hence, only less-chunk boundaries need to be altered. The parameter having the highest effect on the performance is the finger printing algorithm. One of the widely used algorithms for variable-size chunking is Rabin's algorithm.

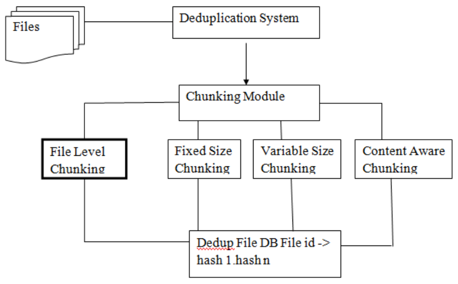


Fig2.1 Different chunking module structures

III. SYSTEM ANALYSIS

Data deduplication systems, the private cloud are involved as a proxy to allow data owner/users to securely perform duplicate check with differential privileges. Such architecture is practical and has attracted much attention from researchers. The data owners only outsource their data storage by utilizing public cloud while the data operation is managed in private cloud.Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, de duplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy.Deduplication can take place at either the file level or the block level. For file level deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files. Identical data copies of different users will lead to different cipher texts, making de duplication impossible.

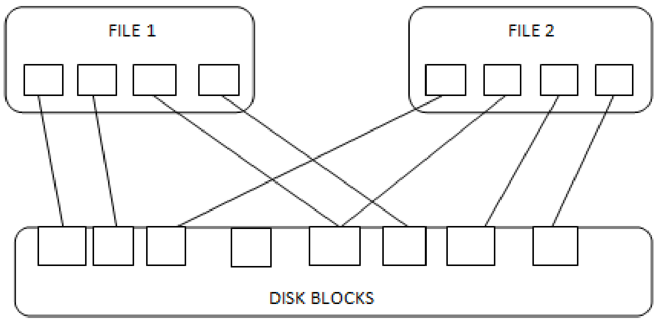


Fig2.2 File Level Deduplication

We enhance our system in security. Specifically, we present an advanced scheme to support stronger security by encrypting the file with differential privilege keys. In this way, the users without corresponding privileges cannot perform the duplicate check. Furthermore, such unauthorized user's cannot decrypt the cipher text even collude with the S-CSP. Security analysis demonstrates that our system is secure in terms of the definitions specified in the proposed security model.Convergent encryption has been proposed to enforce data confidentiality while making deduplication feasible. It encrypts/decrypts a data copy with a convergent key, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the cipher text to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipher text. To prevent unauthorized access, a secure proof of ownership protocol is also needed to provide the proof that the user indeed owns the same file when a duplicate is found.

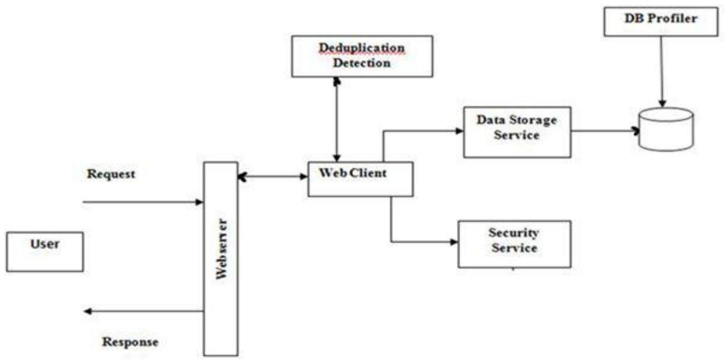


Fig 2.3 Proposed Architecture

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

IV. SYSTEM DESIGN

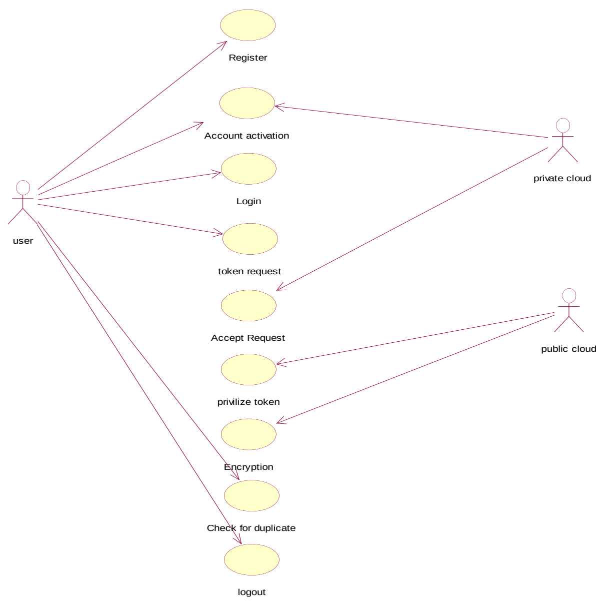
System Design involves identification of classes their relationship as well as their collaboration. In objector, classes are divided into entity y classes and control classes. The Computer Aided Software Engineering (CASE) tools that are available commercially do not provide any assistance in this transition. CASE tools take advantage of Meta modeling that is helpful only after the construction of the class diagram. In the FUSION method some object-oriented approach likes Object Modeling Technique (OMT), Classes, and Responsibilities. Collaborators (CRC), etc, are used. Objector used the term" agents" to represent some of the hardware and software system. In Fusion method, there is no requirement phase, where a user will supply the initial requirement document. Any software project is worked out by both the analyst and the designer. The analyst creates the user case diagram. The designer creates the class diagram. But the designer can do this only after the analyst creates the use case diagram. Once the design is over, it is essential to decide which software is suitable for the application.

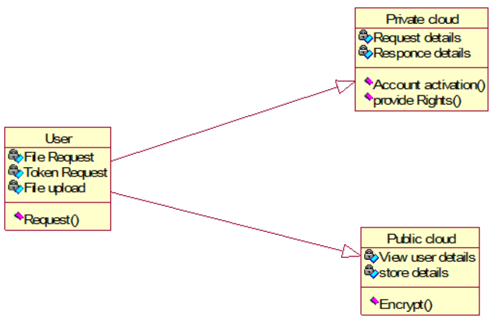
4.1 UML DIAGRAM OF THE PROJECT:

UML is a standard language for specifying, visualizing, and documenting of software systems and created by Object Management Group (OMG) in 1997.There are three important type of UML modeling are Structural model, Behavioral model, and Architecture model. To model a system the most important aspect is to capture the dynamic behavior which has some internal or external factors for making the interaction. These internal or external agents areknown as actors. It consists of actors, use cases and their relationships. In this fig we represent the Use Case diagram for our project.

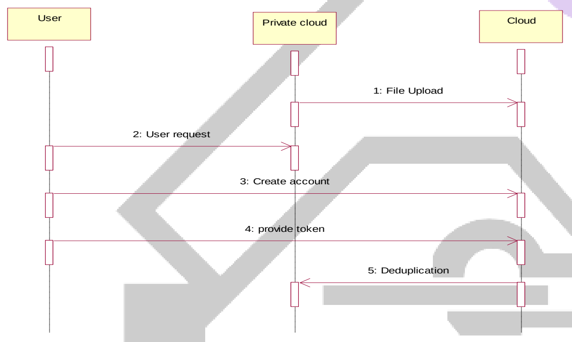
4.1.1 Use case diagram:

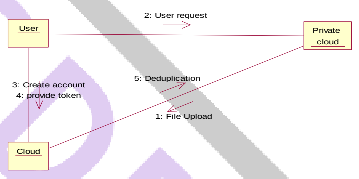
A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.





Class Diagram





Sequence Diagram

Collaboration Diagram

V. SYSTEM TESTING

5.1 TESTING OBJECTIVES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.2 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

5.3 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more

Concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

5.4 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is Focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.5 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.5.1 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.5.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.6 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.
Pages must be activated from the identified link.
The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format
No duplicate entries should be allowed
All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

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:

All the test cases mentioned above passed successfully. No defects encountered.

VI. CONCLUSION

In this paper, the notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

REFERENCES

[1] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In USENIX Security Symposium, 2013.
[2] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In EUROCRYPT, pages 296–312, 2013.
[3] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.
[4] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In CRYPTO, pages 162–177, 2002.
[5] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In Workshop on Cryptography and Security in Clouds (WCSC 2011), 2011.
[6] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.
[7] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In USENIX Security Symposium, 2013.
[8] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In EUROCRYPT, pages 296– 312, 2013.
[9] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.
[10] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In CRYPTO, pages 162–177, 2002.
[11] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In Workshop on Cryptography and Security in Clouds (WCSC 2011), 2011.
[12] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In ICDCS, pages 617– 624, 2002.
[13] D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.
[14] S. Halevi, D. Harnik, B. Pinkas, and A. ShulmanPeleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, ACM Conference on Computer and Communications Security, pages 491–500. ACM, 2011.
[15] J. Li, X. Chen, M. Li, J. Li, P. Lee, andW. Lou. Secure deduplication with efficient and reliable convergent key management. In IEEE Transactions on Parallel and Distributed Systems, 2013.