International Journal of Advanced Technology in Engineering and Science
Vol. No.7, Issue No. 03, March 2019
www.ijates.com
ijates
ISSN 2348 - 7550

Efficient Client-Side Deduplication of Encrypted Data with Public Auditing in Cloud Storage

AHSOKKUMAR. K [GUIDE]

SUGAVATHI.S BE -Computer Science And Engineering,Sengunthar Engineering College.

RAMYA.C BE-Computer Science And Engineering,Sengunthar Engineering College.

RAMYA.A BE-Computer Science And Engineering,Sengunthar Engineering College.

ABSTRACT

At present, there is a considerable increase in the amount of data stored in storage services,along with dramatic evolution of networking techniques. In storage services with huge data, the storageservers may want to reduce the volume of stored data, and the clients may want to monitor the integrity oftheir data with a low cost, since the cost of the functions related to data storage increase in proportion to the size of the data. To achieve these goals, secure deduplication and integrity auditing delegation techniqueshave been studied, which can reduce the volume of data stored in storage by eliminating duplicated copiesand permit clients to efficiently verify the integrity of stored files by delegating costly operations to a trustedparty, respectively. So far many studies have been conducted on each topic, separately, whereas relativelyfew combined schemes, which supports the two functions simultaneously, have been researched. In thispaper, we design a combined technique which performs both secure deduplication of encrypted data andpublic integrity auditing of data. To support the two functions, the proposed scheme performs challenge-response protocols using the BLS signature based homomorphic linear authenticator. We utilize a third partyauditor for performing public audit, in order to help low-powered clients. The proposed scheme satisfies allthe fundamental security requirements. We also propose two variances that provide higher security andbetter performance.

INDEX TERMS:Cloud storage, Cryptography, Data security, Information security, Public audit, Securededuplication

INTRODUCTION

IN cloud storage services, clients outsource data to aremote storage and access the data whenever they needthe data. Recently, owing to its convenience, cloud storageservices have become widespread, and there is an increase inthe use of cloud storage services. Well-knowncloud servicessuch as Dropbox and iCloud are used by individuals andbusinesses for various applications. A notable change ininformation-based services that has happened recently is thevolume of data used in such services due to the dramatic evo-lution of network techniques. For example, in 5G networks,gigabits of data can be transmitted per second, which meansthat the size of data that is dealt by cloud storage servicewill increase due to the performance of the new networkingtechnique. In this viewpoint, we can characterize the volumeof data as a main feature of cloud storage services. Many service providers have already prepared high resolution contentsfor their service to utilize faster networks. For secure cloudservices in the new era, it is important to prepare suitablesecurity tools to support this change.Larger volumes of data require higher cost for managingthe various aspects of data, since the size of data influencesthe cost for cloud storage services. The scale of

storageshould be increased according to the quantity of data to be volume. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission.This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. In this viewpoint, it is desirable for storage servers toreduce the volume of data, since they can increase their profitby reducing the cost for maintaining storage. On the otherhand, clients are mainly interested in the integrity of theirdata stored in the storage maintained by service providers.To verify the integrity of stored files, clients need to performcostly operations, whose complexity increases in proportionto the size of data. In this viewpoint, clients may want to verify the integrity with a low cost regardless of the size ofdata. Owing to the demands of storage servers and clients,many researches on this topic are available in the literature.To reduce the volume of data, deduplication has to be per-formed in servers so that the storage space efficiency can beimproved by removing duplicated copies. In these concept of proofs of ownership (PoW). In Bellare et al.formalized a class of message-locked encryptions includingan existing convergent encryption (CE), and presented a newdeduplication technique called DupLESS which is the firstdeduplication mechanism that can ensure semantic security.When clients use cloud storage services, the integrity ofstored data is the most important requirement. In other words,clients want to be guaranteed about the integrity of their datain the cloud. In cloud storage services, we cannot excludethe possibility of weak cloud servers, which are vulnerableto internal and external security threats. In the case of dataloss due to some incident, weak servers may try to hide thefact that they lost some data, which were entrusted by theirclients. More seriously, servers delete rarely accessed users'data in order to increase the profit. Therefore, it is a naturalrequirement of clients to periodically check the current stateof their data. To do this in practice, we need a way toefficiently check the integrity of data in remote storage. secure deduplication and integrity auditing are fundamental functions required in cloud storage services. Hence, individual researches have been actively conducted on these twotopics. However, relatively few studies have been conductedfor designing a combined scheme that can support these twofunctions at the same time. The fundamental goal of the de-sign of a combined model is to guarantee less overhead thana trivial combination of existing schemes. In particular, thegoal of this paper is to improve the cost of both computationand communication.In this paper, we design a new scheme for secure andefficient cloud storage service. The scheme supports bothsecure deduplication and integrity auditing in a cloud envi-eonment. In particular, the proposed scheme provides securededuplication of encrypted data. Our scheme performs PoWfor secure deduplication and integrity auditing based on thehomomorphic linear authenticator (HLA), which is designedusing BLS signature.

LITERATURE REVIEW

1)CONFUCIOUS: A TOOL SUPPORTING COLLABORATIVE SCIENTIFIC WORKFLOW COMPOSITION

A research is an enabling collaboration technique in the aspect of collaboration provenance management and reproducibility. Based on scientific collaboration ontology, it proposed a service-oriented collaboration model supported by a set of collaboration primitives and patterns. The collaboration protocols are then applied to support effective concurrency control in the process of collaborative workflow composition. It also reports the design and development of confucious, a service-oriented collaborative scientific workflow composition tool that extends an open-source, single-user environment.

TECHNOLOGY: Floor granting algorithm,Locking Algorithm

DISADVANTAGE **:** Do not support scientific workflow application.

2)SECURE AND PRACTICAL OUTSOURCING OF LINEAR PROGRAMMING IN CLOUD COMPUTING

This system investigates secure outsourcing of widely applicable Linear Programming (LP) computations. In order to achieve practical efficiency, this mechanism design explicitly decomposes the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. The resulting flexibility allows us to explore appropriate security or efficiency tradeoff via higher-level abstraction of LP computationsthan the general circuit representation.

TECHNOLOGY **:** RS algorithm

DISADVANTAGE **:** DES algorithm used to share the files, with high Cost.

3) REAL TIME TASKS ORIENTED ENERGY-AWARE SCHEDULING IN VIRTUALIZED CLOUDS

Energy-aware scheduling algorithms developed for clouds are not real-time task oriented, thus lacking the ability of guaranteeing system schedule ability. To address this issue, this system used a novel rolling-horizon scheduling architecture for real-time task scheduling in virtualized clouds. Based on its scheduling architecture, it develop a novel energy-aware scheduling algorithm named EARH for real-time, aperiodic, independent tasks. The EARH employs a rolling-horizon optimization policy and can also be extended to integrate other energy-aware scheduling algorithms. Furthermore, it propose two strategies in terms of resource scaling up and scaling down to make a good trade-off between task's schedule ability and energy conservation.

TECHNOLOGY **:**Energyaware scheduling pseudocode, Pseudocode
of energy-efficient scheduling

DISADVANTAGES **:** Not improve the scheduling quality, high energy.

4) MEETING DEADLINES OF SCIENTIFIC WORKFLOWS IN PUBLIC CLOUDS WITH TASKS REPLICATION

The elasticity of Cloud infrastructures makes them a suitable platform for
execution of deadline-constrained workflow applications, because resources available to the application can be dynamically increased to enable application speed up. Existing research in execution of scientific workflows in Clouds either try to minimize the workflow execution time ignoring deadlines and budgets or focus on the minimization of cost while trying to meet the application deadline. However, they implement limited contingency strategies to correct delays caused by underestimation of tasks execution time or fluctuations in the delivered performance of leased public Cloud resources.

TECHNOLOGY : TheBioinformatics,PC(PartialCriticalPath)technique,PBTSalgorithm

DISADVANTAGE: Not support scientific workflow application, underestimated execution
time.

5) COST-DRIVEN SCHEDULING OF GRID WORKFLOWS USING PARTIAL CRITICAL PATHS

One of the most challenging problems in utility Grids is workflow scheduling, i.e., the problem of satisfying the QoS (Quality of Service) of the users as well as minimizing the cost of workflow execution. In this system, it propose a new QoS-based workflow scheduling algorithm based on a novel concept called Partial Critical Paths (PCP), that tries to minimize the cost of workflow execution while meeting a user-defined deadline. The PCP algorithm has two phases: in the deadline distribution phase, it recursively assigns sub deadlines to the tasks on the partial critical paths ending at previously assigned tasks, and in the planning phase it assigns the cheapest service to each task while meeting its sub deadline.

TECHNOLOGY: Partial critical path algorithm, Parents assigning,

Path assigning, Optimized path algorithm.

DISADVANTAGE **:** Not approximate execution and data transmission.

Advanced Encryption Standard (AES)

Definition

The Advanced Encryption Standard (AES) is an encryption algorithm for securing sensitive but unclassified material by U.S. Government agencies and, as a likely consequence, may eventually become the de facto encryption standard for commercial transactions in the private sector. (Encryption for the US military and other classified communications is handled by separate, secret algorithms.)In January of 1997, a process was initiated by the National Institute of Standards and Technology (NIST), a unit of the U.S. Commerce Department, to find a more robust replacement for the Data Encryption Standard (DES) and to a lesser degree Triple DES. The specification called for a symmetric algorithm (same key for encryption and decryption) using block encryption (see block cipher) of 128 bits in size, supporting key sizes of 128, 192 and 256 bits, as a minimum. The algorithm was required to be royalty-free for use worldwide and offer security of a sufficient level to protect data for the next 20 to 30 years. It was to be easy to implement in hardware and software, as well as in restricted environments (for example, in a smart card) and offer good defenses against various attack techniques.The entire selection process was fully open to public scrutiny and comment, it being decided that full visibility would ensure the best possible analysis of the designs. In 1998, the NIST selected 15 candidates for the AES, which were then subject to preliminary analysis by the world cryptographic community, including the National Security Agency. On the basis of this, in August 1999, NIST selected five algorithms for more extensive analysis. These were:

MARS, submitted by a large team from IBM Research

RC6, submitted by RSA Security

Rijndael, submitted by two Belgian cryptographers, Joan Daemen and Vincent Rijmen

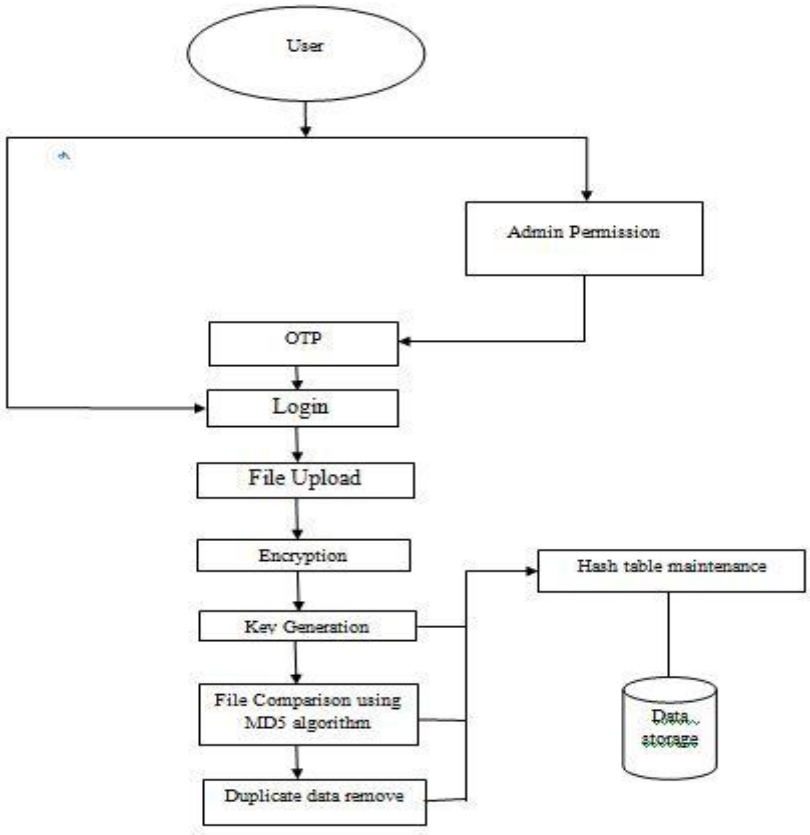
Serpent, submitted by Ross Andersen, Eli Biham and Lars Knudsen

Twofish, submitted by a large team of researchers including Counterpane's respected cryptographer, Bruce Schneier

Implementations of all of the above were tested extensively in ANSI C and Java languages for speed and reliability in such measures as encryption and decryption speeds, key and algorithm set-up time and resistance to various attacks, both in hardware- and software-centric systems. Once again, detailed analysis was provided by the global cryptographic community (including some teams trying to break their own submissions). The end result was that on October 2, 2000, NIST announced that Rijndael had been selected as the proposed standard. On December 6, 2001, the Secretary of Commerce officially approved Federal Information Processing Standard (FIPS) 197, which specifies that all sensitive, unclassified documents will use Rijndael as the Advanced Encryption Standard.Also see cryptography, data recovery agent (DRA)

RELATED GLOSSARY TERMS: RSA algorithm (Rivest-Shamir-Adleman), data key, greynet (or graynet), spam cocktail (or anti-spam cocktail), fingerscanning (fingerprint scanning),munging, insider threat, authentication server, defense in depth, nonrepudiation

SYSTEM



ARCHITECTURE:

Big Bang

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

A type of Big Bang Integration testing is called **Usage Model testing**. Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use.

Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.

For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

Testing

The various levels of testing are

White Box Testing

White-box testing (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during

integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code.

These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working

as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

Integration testing. White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

White-box testing's basic procedures involve the understanding of the source code that you are testing at a deep level to be able to test them. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analysed for test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

1. Input, involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.
2. Processing Unit, involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
3. Output, prepare final report that encompasses all of the above preparations and results.

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration,system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well

Test procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

Unit testing

In computer programming, **unit testing** is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper)to being formalized as part of build automation.

Testing will not catch every error in the program, since it cannot evaluate every execution path in any but the most trivial programs. The same is true for unit testing. Additionally, unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance).

Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. In order to guarantee correct behaviour for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behaviour.

Software testing is a combinatorial problem. For example, every Boolean decision statement requires at least two tests: one with an outcome of "true" and one with an outcome of "false". As a result, for every line of code written, programmers often need 3 to 5 lines of test code.

This obviously takes time and its investment may not be worth the effort. There are also many problems that cannot easily be tested at all – for example those that are nondeterministic or involve multiple threads. In addition, code for a unit test is likely to be at least as buggy as the code it is testing. Fred Brooks in The Mythical Man-Month quotes: *never take two chronometers to sea. Always take one or three.* Meaning, if two chronometers contradict, how do you know which one is correct?

Another challenge related to writing the unit tests is the difficulty of setting up realistic and useful tests. It is necessary to create relevant initial conditions so the part of the application being tested behaves like part of the complete system. If these initial conditions are not set correctly, the test will not be exercising the code in a realistic context, which diminishes the value and accuracy of unit test results.

To obtain the intended benefits from unit testing, rigorous discipline is needed throughout the software development process. It is essential to keep careful records not only of the tests that have been performed, but also of all changes that have been made to the source code of this or any other unit in the software. Use of a version control system is essential. If a later version of the unit fails a particular test that it had previously passed, the version-control software can provide a list of the source code changes (if any) that have been applied to the unit since that time.

It is also essential to implement a sustainable process for ensuring that test case failures are reviewed daily and addressed immediately if such a process is not implemented and ingrained into the team's workflow, the application will evolve out of sync with the unit test suite, increasing false positives and reducing the effectiveness of the test suite.

Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs.[7]

Functional testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes *what* the system does.

Functional testing differs from system testing in that functional testing "*verifies* a program by checking it against ... design document(s) or specification(s)", while system testing "*validate* a program by checking it against the published user or system requirements" (Kane, Falk, Nguyen 1999, p. 52).

Functional testing typically involves five steps .The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications
2. The determination of output based on the function's specifications
3. The execution of the test case
4. The comparison of actual and expected outputs.

Performance testing

In software engineering, **performance testing** is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

Load testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set

duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

Stress testing

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

Soak testing

Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

Spike testing

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

Configuration testing

Rather than testing for performance from the perspective of load, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behaviour. A common example would be experimenting with different methods of load-balancing.

Isolation testing

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

Integration testing

Integration testing (sometimes called **integration and testing**, abbreviated **I&T**) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in

larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up. Other Integration Patterns are: Collaboration Integration, Backbone Integration, Layer Integration, Client/Server Integration, Distributed Services Integration and High-frequency Integration.

CONCLUSION

The newly proposed system is complete system to securely outsource log records to a cloud provider. In this work, find out the challenges for a secure cloud based log management service. The attackers use below three steps to hack. First, the attacker can intercept any message sent over the Internet. Second, the attacker can synthesize, replicate, and replay messages in his possession and the attacker can be a legitimate participant of the network or can try to impersonate legitimate hosts. It implement how to store secure log file in cloud and that file we can change read, write, delete, upload and download. It can implement AES algorithm that uses for log monitor and log generator .One of the unique challenges is the problem of log privacy that arises when we outsourced log management to the cloud. Log information in this case should not be casually linkable or traceable to their sources during storage, retrieval and deletion. It provided anonymous upload, retrieve and delete protocols on log records in the cloud using the Tor network. The protocols that it developed for this purpose have potential for usage in many different areas including anonymous publish-subscribe.

FUTURE ENHANCEMENT

The basic idea of secure De-duplication services can be implemented given additional security features insider attacker on De-duplication and outsider attacker by using the detection of masquerade activity which means unknown person stolen and damage the data. So we confusion of the attacker and the additional costs incurred to

distinguish real from fake information added, and the deterrence effect which, although hard to measure, plays a significant role in preventing from the attackers, that will harmful for our data

.

REFERENCE:

1) Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song,Provable data possession at untrusted stores," in Proc. of the 14th ACM conference on Computer and communications security (CCS'07), Alexandria, Virginia, USA, 2007, pp. 598–609.

[2] G. Ateniese, R. Di Pietro, L.V. Mancini and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of the 4th international conference on Security and privacy in communication netowrks (SecureComm'08), Istanbul, Turkey, 2008, pp. 1–10.

[3] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing," Journal of Cryptology, vol. 17, no. 4, pp. 297–319, Sept. 2004.

[4] Y. Dodis, S. Vadhan and D. Wichs, "Proofs of retrievability via hardness amplification," in Proc. of the 6th Theory of Cryptography Conference on Theory of Cryptography (TCC'09), San Francisco, CA, USA, 2009, pp. 109–127.

[5] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," NIST, USA, No. NIST-SP-800-38A., 2001.

[6] A. Juels and B.S. Kaliski Jr, "Pors: proofs of retrievability for large files,"in Proc. of the 14th ACM conference on Computer and communicationssecurity (CCS'07), Alexandria, Virginia, USA, 2007, pp. 584–597.

[7] S. Keelveedhi and M. Bellare and T. Ristenpart, "DupLESS: server-aided encryption for deduplicated storage," in Proc. of the 22nd USENIXSecurity Symposium (USENIX Security 13), Washington, D.C. USA,2013, pp. 179–194.

[8] J. Li, J. Li, D. Xie and Z. Cai, "Secure auditing and deduplicating data incloud," IEEE Transactions on Computers, vol. 65, no. 8, pp. 2386–2396,Aug. 2016.

[9] X. Liu, W. Sun, H. Quan, W. Lou, Y. Zhang and H. Li, "Publicly verifiableinner product evaluation over outsourced data streams under multiplekeys," IEEE Transactions on Services Computing, vol. 10, no. 5, pp. 826-838, Sept.-Oct. 2017.

[10] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. Ofthe 14th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology – ASIACRYPT2008, Melbourne, Australia, 2008, pp. 90–107.

[11] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling public auditabilityand data dynamics for storage security in cloud computing," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 5, pp. 847–859,Dec. 2011.

[12] T. Y. Youn, K. Y. Chang, K. R. Rhee and S. U. Shin, "Public Auditand Secure Deduplication in Cloud Storage using BLS signature," Research Briefs on Informaiton& Communication Technology Evolution(ReBICTE), vol. 3, article no. 14, pp. 1-10, Nov. 2017.

[13] J. Yuan and S. Yu, "Proofs of retrievability with public verifiability andconstant communication cost in cloud," in Proc. of the 2013 internationalworkshop on Security in cloud computing, Hangzhou, China, 2013, pp.19–26.

[14] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditingwith deduplication," in Communications and Network Security (CNS),2013 IEEE Conference on, National Harbor, MD, USA, 2013, pp. 145-153.