Secure File Allocation in Cloud Storage Using Audit Services and Deduplication

Burlu Revathi[1], N.P.Patnayak.M[2]
#1. M.Tech Scholar (CSE) in Department of Computer Science Engineering,
#2. Assistant Professor, Department of Computer Science Engineering,Viswanadha Institute of Technology and Management, Visakhapatnam, A.P, India

Abstract:-

Data de-duplication is a method of getting rid of duplicate copies of data and has typically been used in cloud storage to reduce the amount of data stored and transferred. However, despite the likelihood that a document is requested by a huge number of clients, there is only one copy of every file stored in the cloud. De-duplication framework hence improves storage use while lowering constant quality. Additionally, when clients outsource their sensitive data to the cloud, security for that data is put to the test. In order to address the aforementioned security issues, the main goal of this study is to codify the concept of an adequate reliable deduplication architecture. In this paper, new distributed deduplication frameworks with improved unwavering quality are provided, in which the data chunks are sent via various cloud servers. Cloud lessens the burden of storing and maintaining the enormous amounts of data. The issues of unwavering quality and security concerns are brought up by external cloud storage. Deduplication and integrity auditing problems are partially caused by the current framework. The main objective of this study is to achieve data integrity and deduplication in order to maintain the security and soundness of our system. With the aid of a guide lessen system, calculating times for data transferring and downloading were slashed in earlier work. Clients' private information is typically more safe, thus this uses encryption to increase security.

Keywords: Public Verification Parameter, Cloud Service Provider, Auditor, Cloud Storage Service, Proof of Data Possession. Introduction to SecCloud, Deduplication, and I.

1.Introduction

The enormous advancement in the cloud computing industry offers consumers a wide range of benefits, including data accessibility from topographically circulated areas, consistent quality, quick sending, data documentation, disaster recovery, and strong security for reinforcement. Despite the enormous benefits provided by cloud storage, "Integrity auditing" and "Deduplication" are some real security risks. We focus on the aforementioned difficulties in this study and present two secure topologies, namely "SecCloud" and "SecCloud+." On the Internet, computation and capacity resources are provided by cloud computing. A growing amount of data is being stored in the cloud and is being given by customers to specific benefits, indicating unique rights to data storage. Dealing with the exponential growth of the ever-growing volume of data has become a fundamental test. According to the IDC cloud report 2014, businesses in India are continuously transitioning from on-premises legacy to different types of cloud. Although the process is ongoing, it has started by shifting some application workloads to the cloud [1]. De-duplication [2] has been a remarkable process that has recently become increasingly common, allowing for flexible management of stored data in cloud computing. De-duplication is a particular data compression technique that reduces storage space and transfer capacity in cloud storage. De-duplication replaces repetitive data with a pointer to the interesting data duplicate so that only one unique instance of the data is actually on the server. Either at the record level or the piece level, deduplication can take place. Customers' concerns about security and safety arise since data are defenceless against attacks from both insiders and outsiders. Classification, integrity checking, and getting to handle both of the assaults' components should be properly authorised. Normal encryption does not support deduplication. Client scrambles their documents using their unique encryption key; unique figure content would appear even for identical documents. In this way, data deduplication and conventional encryption are incompatible. A popular technique to combine the capacity-saving of de-duplication to maintain classification is joined

encryption [3]. In focalized encryption, the duplicate data is encoded using a key that can be deduced from the data's own hash. A data duplication is encoded and decoded using this combined key. Clients hold the keys after key erasure and data encryption, and they communicate the figure content to the cloud. Indistinguishable data duplicates will result in the same concurrent key and a comparable figure content since encryption is deterministic. This enables the cloud to do figure writing deduplication. The corresponding data owners must use their combined keys to decode the figure writings. A recognised de-duplication technique is differential approval copy check, in which each customer receives a set of benefits at the introduction of the framework. This list of benefits identifies the types of clients who can access the papers and do a copy check. hiding the execution's stage and delicate details Cloud computing is the management of endless virtualized assets for the clients. Without further ado, the clients were given access to high-capacity storage and massively parallel asset processing through the use of the cloud, both of which were generally inexpensive. However, the subject concerns cloud clients with a variety of advantages. The most challenging aspect of managing a cloud data storage architecture is storing data on the cloud.

II. Companion Work

There are a number of remarkable techniques used as part of various auditing systems that provide integrity beware of cloud. [A] Mechanisms used for Cloud data integrity check 1. HLA-Based Approach This system was explained by Chandinee Saraswathy et al in [2], which uses straight mix for validation and aids in auditing without recovering data pieces and checking the integrity of the data. Only confirmed meta data is checked by HLA. It verifies data via a direct blending of the separate data elements to check the square's integrity. This feature enables efficient data auditing while consuming only constant transmission capacity, but it has the downside of being time-consuming due to the use of a straight mix for confirmation. 2. Minimal Retrievability Proofs Hovav Shacham and Brent Watersy [3] presented a framework for verifying retrievability. In this integrity check methodology, the data storage centre provides proof to the auditor that it is indeed storing the majority of a customer's data. Here, two homomorphic authenticators have been made clear. The principal authenticator, which depends on PRFs, provides a proof-of-retrievability plan that is secure in the traditional paradigm. The second is based on BLS marks [4], which provide an open fluctuation secure proof-of-retrievability plan in the arbitrary prophet display. Structures that have been clarified allow for debate over the frameworks' invulnerability, extractability, and retrievability with these components in the context of discrete cryptographic, combinatorial, and coding systems. 3 PDP at Untrusted Stores: Provable Data Possession A paradigm was put up by Giuseppe Ateniese et al. [5] in light of proven data ownership (PDP). This PDP is used to verify that the server is preparing the initial data to be made available in the cloud without retrieving it or reviewing its content. This model examines arbitrary configurations of server-sourced data components to generate probabilistic ownership verification. This cost-effective method lowers the cost of I/O. Consistent Data Possession Forward Error Correcting codes (FEC) are coordinated into Provable Data Possession by this component [6]. (PDP). FEC code is used to encrypt the first record, and PDD is then connected to the encoded document rather than the original record. Here, it has two benefits: first, it prevents debasement of a small portion of the document because changes inside the document are also recognised because FEC code is used, and second, it anticipates debasement of a complete record when there is a change to the squares. 5. Security Maintaining Public Auditing The Privacy Preserving Public Auditing approach was introduced by Cong Wang [7]. Here, open auditing enables the client and auditor to verify the accuracy of the outsourced data stored in the cloud, and privacy-preserving technology enables the auditor to conduct the audit without requesting any data. The auditor is then prepared to review the data while maintaining cloud data security. In order to prevent the auditor from learning anything about the data content present on the cloud server during the productive auditing process, they have used two systems: the homomorphic straight authenticator and the irregular covering. This relieves the cloud client from the tedious and potentially expensive auditing undertaking and also prevents the clients from fear of the outsourced data spillage.

III. Problem formulation

The existing system has issues with auditing integrity as a result of the large amount of data on cloud servers and the user's demand for correct data retrieval from the cloud rather than complete data, according to the problem statement. This issue was resolved in the system that was suggested. Due to the growing number of duplicate files in the cloud, another issue with deduplication of data has emerged. These duplicate files cause storage space to be exceeded as well as increase computation time.

IV. Overview of Secure Deduplication

Clients start the ingestion process by splitting a file into a series of pieces in both the anonymous and authenticated models. A content-based chunking process, which creates chunks depending on the file's contents, is frequently used to achieve this. This method has the advantage of matching shared material between files even when that content does not exist at the multiple of a predetermined, fixed offset [25]. The algorithm chooses chunks based on a sliding window of width w that is moved over the file and a threshold value A. A fingerprint of the window's contents, Fk,k+w1, of each place k in the file is computed [28]. Fk,k+w1 > A, in which case k is chosen as a chunk boundary. The outcome is a collection of chunks of varying sizes, where the data's content determines the boundary between chunks. On the client, file chunking and encryption take place. The advantages of carrying out these operations on the client as opposed to the server are several. First of all, it lessens the amount of processing required on the server. Additionally, by encrypting client-side chunks, data is never sent in the open, lessening the impact of several passive external attacks. Third, since the server would not have to store the encryption keys, a powerful, hostile insider would not have access to the data's plaintext. Convergent encryption, which was introduced in the Farsite system [10], is used by clients to encrypt chunks. This method uses a cryptographic hash of the plaintext as the key; Farsite and our system utilise an encryption key that is deterministically derived from the plaintext information to be encrypted. No matter who performs the encryption, a particular plaintext always yields the same ciphertext since identical plaintexts cause the employment of identical keys. K = chunk(hash) (6) This method has a lot of benefits over alternative methods. The amount of storage space saved through deduplication would be significantly decreased if each user encrypted using his or her own key, as we have demonstrated in Section 3, because the same chunk encrypted using two separate keys would produce different ciphertext (with very high probability). Second, there is a key sharing issue when attempting to share a random key among many user accounts. Thirdly, a user who is unaware of the value of the data's plaintext cannot create the key and, as a result, cannot decipher the plaintext from the ciphertext. In contrast to a method where the server encrypts the data, even a root level administrator does not have access to a chunk's plaintext value without the key, making this aspect particularly crucial. As stated in its initial explanation [10], this approach's main security drawback is that it exposes certain information. Convergent encryption, in particular, shows if two ciphertext strings decrypt to the same plaintext value. Information leakage is part of the sacrifice required to achieve space-efficiency through deduplication, yet it is vital in systems that use deduplication as it enables a system to delete duplicate plaintext data chunks while only seeing the ciphertext. An identification must be given to each piece of ciphertext. The encrypted chunk's hash value, also known as content-based naming, is used in our system to uniquely identify each chunk. e(hash(chunk),chunk) = chunk id (7) The hash of the hash of the plain-text chunk can be used as an alternative to the hash of the encrypted chunk; in this case, the encryption key's hash serves as the chunk identifier. This strategy has a lot of appealing features. Performance is enhanced first. In both strategies, the user generates two hashes: a key hash and an identification hash. Performing two chunk hashes will cost more than performing a chunk hash and a key hash, assuming that key lengths are shorter than chunk lengths. Second, rather than preserving both the key and the identifier, the file to chunk map just has to maintain the key if the identification can be deduced from the key. The chunk storage cannot validate the accuracy of the chunk's content-based identifier, which is a significant disadvantage of utilising the hash of the key as the identifier. Unverified chunk signatures enable the use of targeted collision attacks, as was discussed in Section 3.2. The chunk storage contains the encrypted pieces themselves. The information required to find the

appropriate storage device can also be included in the chunk list in a distributed storage paradigm where there may be numerous chunk stores. Alternately, using the identifier of the chunk, deterministic placement techniques can be used to find the proper storage devices.

V. ANALYSIS OF SECURITY

The two secure deduplication models that we have described have been evaluated in order to show that the system is secure in the face of a number of potential threats. We start by looking at the assaults that an outside enemy might launch against the system. Second, we look at security flaws that might arise from an insider with bad intentions having access to all raw data, such a system administrator with root-level access. Third, we look at the security ramifications that result from compromised system keys.

External Opponents

A system must be able to stop information from leaking to an outside attacker in order to be deemed secure. An attacker who intercepts communications between system users would be an example of a passive adversary. A foe that modifies or sends messages is an active example. The passive attacker issue is greatly reduced in both the authenticated and external models by having the client handle the chunking and encryption. As a result, unencrypted data is never sent in clear. The anonymous model does not specify how the keys can be shared securely, but it presupposes that this is possible. Creating a secure protocol for this technique might be a future research focus. Since all information exchanged between players is encrypted, the risk from an active adversary is that communications could be altered. For instance, in the fundamental models we've shown, a chunk could be stopped on its way to the chunk store and changed. Despite the fact that our architecture does not specifically handle these cases, transport layer security (TLS) methods like Secure Sockets Layer can significantly lessen the impact of these assaults. Since the anonymous model aims to conceal the user's identity, an outside adversary may be able to gather some information by determining the source of requests. Our method does not directly solve the problem, as with the man-in-the-middle attacks that were previously discussed. However, solutions like onion routing have addressed this issue and are compatible with our design [12].

Internal Opponents

A secure system must also offer defence against internal attackers, as was covered in Section 3 of this document. In order to accomplish this, we examine an insider's capacity to launch attacks depending on where they are located inside the system and their prospective access levels. A malevolent insider with complete access, as in most systems, might alter or remove any data he wanted, leading to a denial of service attack. Therefore, from a security perspective, our objective is to restrict an insider's capacity to make specific alterations. Limiting these changes involves two different strategies. First, we want to restrict the capacity of an insider to target particular files. Second, we want to restrict an adversary's capacity to make unnoticed changes; replacing a value with junk is typically more noticeable than replacing it with a value that is right in terms of semantics.

Accredited Model

The metadata server does give some information to an internal enemy under the authorised model. First, the file name to inode mapping is accessible to an insider. Second, an internal adversary has access to the inode number for the encrypted map entry. Last but not least, a hostile insider can find out which users and which files each user has access to. A metadata server insider attacker can launch numerous assaults using the information at their disposal. An internal enemy can first remove metadata and deny access to particular users. This attack is undetectable if the client is unaware of the files it should be able to access. Second, the map entry of a different file that the client can access may be returned when a client requests a file. The client's comprehension of the file's contents would determine whether or not this attack is discovered. However, targeted file content modifications demand the adversary to gain the map key. Users grant access according to the current design by sending map keys that have been encrypted with the public key of the authorised user. By doing this, the plaintext key required to access a map entry's details is never disclosed to an insider with harmful intentions. An insider with malevolent intentions could alter the contents of map entries if the system

encrypts map keys. Therefore, concealing the map entry from an inside attacker would be one approach to enhance the system even further. A method like the anonymous model's map references, which, as seen in Figure 4, needs the map key to get the map entry, might be used to achieve this. Finally, it can be inferred that the adversary also has access to portions if a bad insider at the metadata store also distributes capability tickets, as is the case in some systems; a malicious metadata store can easily issue a legitimate capability to itself. But without having access to the map key, the adversary wouldn't be aware of which chunks belong to a specific file and wouldn't have the key to unlock a chunk.

Unknown Model

An inside attacker at the chunk store would be unable to change the data undetected in both the authorised and anonymous models. A user would not be able to request the updated chunk because the chunk name is based on the content, or at the very least, they would be able to discern that the chunk they requested is different from the chunk that was sent to them. Of course, a chunk storage insider might remove pieces or refuse to process requests for chunks. The metadata store in the anonymous paradigm does expose certain data to an internal foe. An insider can first determine which inode numbers correspond to which files. Since the user's symmetric key is required to map inodes to map references, this is not a major problem. But more significantly, since they will all be the same length, an insider may figure out which entries are map references. This is because their payload is always one key rather than a fluctuating list of chunk metadata. Adding some random data at the conclusion of an entry is one approach to prevent the identity of a map-key from being revealed.

Crucial Compromises

Any system that makes use of cryptographic primitives depends heavily on the regulated use of encryption keys to maintain system security. According to Kerckhoff's concept, an adversary's inability to decrypt data gives the system security; it is assumed that the adversary is familiar with the protocols and cryptosystems. Therefore, investigating the effects of key compromises is one method of security system analysis.

Accredited Model

The user's asymmetric key pair is connected to their identity in the authenticated model. Furthermore, it is presumed that an adversary who knows a user's private key also has the user's whole key pair because the public key is readily available via a certificate server. In this case, a malicious user could be able to completely mimic the key's legitimate owner and gain access to all of that user's capabilities. It is advised that authentication demand more information than only the user's key as a defence against this scenario, but this strategy is outside the purview of our model. A less severe information leak is caused by the breach of the map key, the other metadata key in the authorised model. The difficulty of authenticating to the metadata store still exists if an opponent learns the map key. Finally, the old map key can be rendered invalid by using the revocation method to generate a new one. Therefore, even if the map key is compromised, the system is still generally secure. The information leak is also relatively minor if the chunk key, the final key of the authenticated model, is compromised. This is because even if an attacker had the chunk key, they would still need to be able to authenticate to the chunk server, know the chunk identification, and access the plaintext contents.

Unknown Model

The user's private, symmetric key is crucial to the anonymity model's system security. It is safe to presume that every file for which the user has placed a map reference can be accessed by a malicious user if they manage to get their hands on the user's key. They might also try to make the connected list of map entries as long as possible in this situation as another possible attack. Since the anonymous model employs immutable chunks, the file could branch and a new key could be formed. In order to get plaintext data, an adversary only needs the file's inode number if they have access to the map key. This can be done using a brute force attack, presuming that there aren't many inodes. Furthermore, since the system is immutable, even creating a new map key will endanger the old file. The chunk identification is still required, just like in the authenticated approach, in order for an adversary who has the encryption key for the chunk to be able to decrypt the data.

6 Design Objectives

According to the aforementioned concept, the security and performance of our protocol design should be followed to ensure privacy-preserving public auditing for cloud data storage.

Public Audit: It enables auditors to check the accuracy of cloud data on-demand without having to retrieve a copy of the entire data set.

Storage Consistency: The ability of data on a cloud server to withstand auditor auditing without actually storing intact user data.

Batch auditing gives auditors the capacity to handle many auditing delegations from potentially a large number of distinct users at once in a secure and efficient manner.

Lightweight: It makes auditing possible with the least amount of connection and calculation overhead.

7. The suggested system

In contrast to standard clouds, the system designed in this research can attain higher levels of reliability. Here, data secrecy can be achieved with the use of a distributed server; Instead of using a single server, it divided the file into several pieces and stored them on multiple servers..

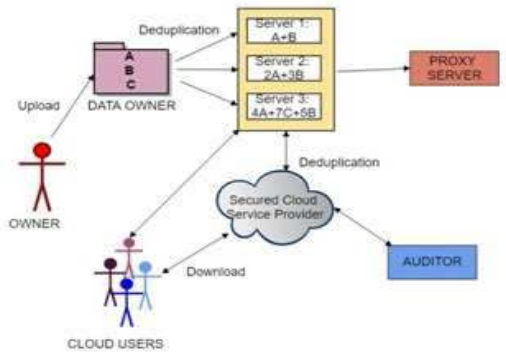

Fig. Proposed System Architecture

This research establishes that the proposed SecCloud system has successfully accomplished file deduplication and integrity auditing. However, this effort forced the capabilities of integrity auditing and deduplication on the plain paper, making it impossible to conceal contents from the server in plain terms. The integrity auditing of encrypted files is done in this work. Given that the system's name is "seccloud system," it provides a key server in this portion of seccloud and distributes keys to its clients there. Here, we're basically adding more functionality to the earlier work that already had an encryption technique. File upload, integrity audit, and proof of ownership methods are all included in Seccloud. The file uploading protocol requires additional steps in order to communicate between the cloud client and the important server. The client in this case needed the convergent key from the key server in order to encrypt the file at the moment of upload.

Key convergence Encryption

Data confidentiality in deduplication is provided through convergent key encryption.

Data will be encrypted using convergent keys, and in order to identify duplicate data, tags that are generated based on the same data will be used to identify it.

• Encrypt(ckF;F): The encryption algorithm accepts the convergent key ckF and file content F as input and outputs the cypher text ctF; • KeyGen(F): The key generation algorithm takes a file content F as input and outputs the convergent key ckF of F;

• Decrypt(ckF; ctF): The decryption algorithm receives the Convergent key ckF and the output file's cypher text, which is ctF.

• TagGen(F): This programme generates the tags for the contents of the file that are designated as TagF using the input from the file's content.

7. Concluding

In this study, distributed deduplication systems are used to increase data dependability while maintaining user data confidentiality without the usage of an encryption technology. To facilitate file-level and precise block-level data deduplication, four constructs were put forth. Consistency and integrity in tag security were attained. The Ramp secret sharing mechanism has been used to create this deduplication system, which has shown that it has a low encoding/decoding overhead compared to the network transmission overhead during typical upload/download activities. The client should consider cloud data security when employing cloud

services. Data security and integrity can be ensured by an auditor. The auditor may be a reliable arbitrator in disputes between the client and the cloud service provider. Over the years, authors have put up a number of different strategies to offer a secure environment for cloud services. Algorithms for encryption and decryption are used to secure user data.

References

[1]. Yan Zhua,b, Hongxin Huc, Gail-Joon Ahnc, Stephen S. Yauc ,"Efficient Audit Service Outsourcing For Data Integrity In Clouds",In The Journal of Systems and Software 85 (2012) .

[2]. Wang.Q, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Audit Ability And Data Dynamics For Storage Security In Cloud Computing", In IEEE Trans. Parallel Distributed Systems, vol. 22, no. 5, pp. 847-859, May 2011.

[3]. Cong Wang, Student Member, IEEE, Sherman S.M. Chow, Qian Wang, Student Member, IEEE, Kui Ren, Member, IEEE, and Wenjing Lou, Member, IEEE, "PrivacyPreserving Public Auditing For Secure Cloud Storage".

[4]. Abhishek Mohta* ,Ravi Kant Sahu,Lalit Kumar Awasthi ,Dept. of CSE, NIT Hamirpur (H.P.) India,"Robust Data Security For Cloud While Using Third Partyauditor"

[5]. Juels.A and J. Burton, S. Kaliski, "Pors: Proofs Of Retrievability For Large Files",In Proc. ACM Conf. Computer and Comm. Security (CCS'07), pp. 584-597, Oct. 2007.

[6]. Ateniese.G, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession At Untrusted Stores" ,In Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 598-609, 2007.

[7]. Jiawei Yuan, Shucheng Yu,"Proofs of Retrievability with Public Verifiability and Constant Communication Cost in Cloud",University of Arkansas at Little Rock ,USA.

[8]. Ezhil Arasu.S, B.Gowri, S.Ananthi ,"Privacy-Preserving Public Auditing In Cloud Using HMAC Algorithm ",In International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-2, Issue-1, March 2013 .

[9]. Shingare Vidya Marshal ,"Secure Audit Service by Using TPA for Data Integrity in Cloud System",In International Journal of Innovative Technology and Exploring Engineering (IJITEE)ISSN: 2278-3075, Volume-3, Issue-4, September 2013.

[10]. Jiawei Yuan,Shucheng Yu "Secure and Constant Cost Public Cloud Storage Auditing with Deduplication", University of Arkansas at Little Rock, USA.

Authors:

BURLU REVATHI holds a B Tech degree in computer science and engineering from nadimpalli satyanarayan Raju institute of technology,sontyam, affiliated to Jawaharlal Nehru technological University, kakinada. She is presently pursuing m tech (CSE) in department of computer science and engineering from viswanadha institute of technology and management, mindivanipalem village, sontyam post, anadapuram mandalam, Visakhapatnam dist, Andhrapradesh.

Mr.**N.P.PATNAYAK.M** working as Assistant Professor, Department of Computer Science Engineering, Viswanadha Institute of Technology and Management, Visakhapatnam, A.P, India.