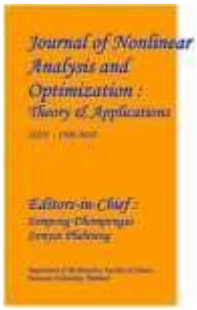
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 16, Issue. 1: 2025
ISSN : **1906-9685**



A Secure Deduplication of Textual Data in Cloud Environments

GANGULA BALAJI[1], Dr. B.V.S Varma[2], Dr. G. SATYANARAYANA[3]
#1 M.Tech Scholar and Department of Computer Science Engineering,
#2 Professor, Department of Computer Science and Engineering, DNR College Of Engineering and Technology, Bhimavaram, AP, India.
#3 Professor, HOD Department of Computer Science and Engineering, DNR College Of Engineering and Technology, Bhimavaram, AP, India.

ABSTRACT:

Present days cloud computing is exceptionally well known and it is spread colossally everywhere throughout the world. Because of expanding vast sum individual data in the cloud condition there are some issue for dealing with the mass data openly cloud space. Data de-duplication is vital system for data pressure which is utilized to wipe out the copy data in the cloud condition. Thus apply the data deduplication method at client side for diminish the excess in there data. Deduplication at Client side system is utilized to recognize copy data as of now at the client and spare the transfer speed of data and transferring chose files to the server. Concurrent encryption is another system which is utilized to better ensure the security of data at client side. Subsequent to encoding data utilizing concurrent key at that point figure is shape, these figure send to cloud before client hold a key. The deterministic idea of encryption, when the indistinguishable data will be transferred with same focalized key and same figure message then deduplication plot keep the copy data. Subsequent to contrasting the data base if coordinate is discovered then just metadata of square store in Database profiler.

Catchphrases: cloud computing; deduplication; copy check; security; concurrent encryption;

I. Introduction

Nowadays, the sensitive advancement of computerized affluence continues raising the interest for arrange limit and extra storage, and furthermore an extending necessity with less cost for the utilization of storage and system transmission capacity keeping in mind the end goal to exchange data. Despite these basic inclinations in sparing assets, deduplication brings various security issues, fundamentally in light of the multi possession challenges. For example, a couple of aggressors target either the usage of data transfer capacity or the security. For example, a client may check whether another client has successfully exchanged a record, by endeavoring to outsource a similar archive to the cloud. Starting late, to direct these worries, various endeavors have been made to propose assorted security models. These diverse schemes are called Proof of Ownership framework (PoW). The storage server is allowed to check a client data possession in light of hash regard. In spite of that the present PoW schemes have kept an eye on various security properties, in any case still require a wary idea of potential assaults, for instance, spillage of data and toxin assaults. So another cryptographic system has been proposed for secure Proof of Ownership (PoW). With a specific end goal to beat security issues in storage, this strategy utilize concurrent encryption and furthermore the Merkle-based Tree. This strategy is effective in giving dynamic sharing between clients. Utilizing the Merkle-based Tree for the data which is scrambled infers an identifier which is remarkable. This identifier permits checking the nearness of similar data in remote cloud servers. Furthermore, accordingly productive data deduplication is accomplished.

II. Related Work

In 2002 J. R. Douceur et al. [5] considered the issue of deduplication in multi-inhabitant condition. The creators proposed the utilization of the merged encryption, i.e., getting keys from the hash of plaintext. At that point M.W.Storer et al. [6] brought up some security issues, and exhibited a security model for secure data deduplication. Notwithstanding, these two conventions center around server-side deduplication and don't consider data spillage settings, against malevolent clients. In this paper [7] M. Bellare et al. Gives either security confirmations or assaults for countless based distinguishing proof and mark schemes characterized either unequivocally or verifiably in existing writing. Hidden these is a system that from one perspective clarifies how these schemes are inferred and then again empowers measured security examinations, subsequently understanding, disentangle, and bind together past work. They likewise dissect a bland legends development that specifically yields personality based recognizable proof and mark schemes without arbitrary prophets. In this paper J. Xu et al. [8] proposed developing requirement for secure cloud storage services and the alluring properties of the concurrent cryptography lead us to join them, subsequently, characterizing an imaginative answer for the data outsourcing security and effectiveness issues. Our answer depends on a cryptographic use of symmetric encryption utilized for enciphering the data document and hilter kilter encryption for Meta data files, because of the most elevated sensibility of this data towards a few interruptions. What's more the Merkle tree properties, this proposition is appeared to help data deduplication, as it utilizes a preverification of data presence, in cloud servers, which is valuable for sparing transfer speed. In addition, our answer is additionally appeared to be impervious to unapproved access to data and to any data revelation amid sharing procedure, giving two levels of access control check. At long last, we trust that cloud data storage security is still brimming with challenges and of vital significance, and numerous examination issues stay to be distinguished. [10],[9] In this paper P. Anderson et al. 2010 [1] proposed an answer here the data which is basic between clients to build the speed of reinforcement and decrease the storage prerequisite specifically reinforcement calculation. Backings client-end per client encryption is important for confidential individual data. This gives the possibility to fundamentally diminish reinforcement times and storage prerequisite. Putting away immense measure of data in PC or PCs causes poor network additionally might be burglary because of equipment disappointment. However Network data transfer capacity can be a contain neck and Backing straightforwardly to a cloud can be expensive are not tended to. Traditional reinforcement arrangements are not appropriate to this condition. So client side deduplication fundamental for confidential individual data. In this paper J.R.Douceur et al. The Farsite appropriated record framework gives accessibility by imitating each document onto different PCs. In the perspective of the way that this replication expends impressive storage space, it is basic to recover utilized space where conceivable. Estimation of more than 500 work area document frameworks demonstrates that almost 50% of all expended space is possessed by copy files. So there is have to show an instrument to recover space from this coincidental duplication to make it accessible for controlled record replication. Our system incorporates concurrent encryption, which empowers copy files to join into the space of a solitary record, regardless of whether the files are encoded with various clients. [3] In this paper M. Mulazzani et al. [4] all through the previous couple of years, a gigantic number of online record storage services have been presented. In the meantime as a few of these services give fundamental usefulness, for example, transferring and recovering files by a particular client, further developed services offer highlights, for example, shared envelopes, constant affiliation, and minimization of data exchanges or unhindered storage space. Reviews of existing document storage services and look at Dropbox, a propelled record storage arrangement, top to bottom. In view of the outcomes they demonstrate that Dropbox is utilized to store copyright-shielded files from a mainstream record sharing system. In this paper M. Bellare et al. Message-Locked Encryption (MLE), where the key under which encryption and decoding are performed is itself gotten from the message. MLE gives an approach to accomplish secure deduplication, an objective right now focused by various cloud-storage suppliers. MLE is a crude of both reasonable and hypothetical concern. [2].

III. BACKGROUND

Deduplication

It is worth mentioning that deduplication can either be file-level [2] or block-level [2]. The latter corresponds to the most common strategy and is also the one to which we refer in this paper. The block size in blockbased deduplication can either be fixed or variable [3]. ***Convergent Encryption Convergent encryption (CE)***

Derives the encryption key from the plaintext. The most common implementations compute key as the hash of the plaintext. Here is a simple example which illustrates how it works: Adam derives the encryption key from her message M such that K = H(M), where H is a cryptographic hash function; she can encrypt the message with this key, hence: C = E(K, M) = E(H(M), M), where E is a block cipher. By applying this technique, two different users with two identical plaintexts will obtain two identical ciphertexts since the encryption key is the same. This allows the cloud storage provider to perform deduplication on such ciphertexts without having any knowledge on the original plain-texts.

Weaknesses of Convergent Encryption

Discussions of vulnerabilities affecting convergent encryption have been presented [3, 10, 9]. As discussed in Section 1, potential malicious cloud providers can perform offline dictionary attacks and discover predictable files. This explains why a strategy is needed to enforce security while retaining benefits offered by deduplication and convergent encryption. **IV. Secure Deduplication Overview**

In both the anonymous and authenticated models, clients begin the ingestion process by transforming a file into a set of chunks. This is often accomplished using a content-based chunking procedure which produces chunks based on the contents of the file. The advantage of this approach is that it can match shared content across files even if that content does not exist at the multiple of a given, fixed offset [25]. The algorithm selects chunks based on a threshold value A and a sliding window of width w that is moved over the file. At each position k in the file, a fingerprint,

$F_{k,k+w-1}$, of the window's contents is calculated [28]. If $F_{k,k+w-1} > A$, then k is selected as a chunk boundary. The result is a set of variable sized chunks, where the boundary between chunks is based on the content of the data. Both file chunking and encryption occur on the client. There are a number of benefits to performing these tasks on the client, as opposed to the server. First, it reduces the amount of processing that must occur on the server. Second, by encrypting chunks on the client, data is never sent in the clear, reducing the effectiveness of many passive, external attacks. Third, a privileged, malicious insider would not have access to the data's plaintext because the server does not need to hold the encryption keys. Clients encrypt chunks using convergent encryption, which was introduced in the Farsite system [10]. Using this approach, clients use an encryption key deterministically derived from the plaintext content to be encrypted; both Farsite and our system use a cryptographic hash of the plaintext as the key. Since identical plaintexts result in the use of identical keys, regardless of who does the encryption, a given plaintext always results in the same ciphertext. K = hash(chunk) (6) Compared to other approaches, this strategy offers a number of advantages. As we have shown in Section 3, if each user encrypted using his own key, the amount of storage space saved through deduplication would be greatly reduced because the same chunk encrypted using two different keys would be would result in different ciphertext (with very high probability). Second, attempting to share a random key across several user accounts introduces a key sharing problem. Third, a user that does not know the data plaintext value cannot generate the key, and therefore cannot obtain the plaintext from the ciphertext. This point is especially important since, in contrast to an approach where the server encrypts the data, even a root level administrator does not have access to a chunk's plaintext value without the key. The primary security disadvantage of this approach, as identified in its original description [10], is that it leaks some information. In particular, convergent encryption reveals if two ciphertext strings decrypt to the same plaintext value. However, this behavior is necessary in systems that use deduplication, since it allows a system to remove duplicate plaintext data chunks while only observing the ciphertext; information leakage is part of the compromise needed to achieve space-efficiency through deduplication. Each ciphertext chunk must be assigned an identifier. In our system, each chunk in the system is identified using the encrypted chunk's hash value, a technique sometimes referred to as contentbased naming. chunk_id = hash(e(hash(chunk),chunk)) (7) An alternative to using

the hash of the encrypted chunk is to use the hash of the hash of the plain-text chunk, i.

e., the hash of the encryption key is the chunk identifier. This approach offers a number of attractive qualities. First, performance is improved. In both approaches the user performs two hashes: a key generation hash, and an identifier generation hash. Assuming that key lengths are smaller than chunk lengths, performing two chunk hashes will be more expensive than a chunk hash and a key hash. Second, if the identifier can be derived from the key, then the file to chunk map only needs to preserve the key, as opposed to the key and the identifier. However, there is a large drawback of using the hash of the key as the identifier: the chunk store cannot verify that the chunk's content-based identifier is correct. As Section 3.2 explained, unverified chunk signatures permit the use of targeted collision attacks. The encrypted chunks themselves are stored within the chunk store. In a distributed storage model, where there may be multiple chunk stores, the chunk list can also include the information needed to locate the correct storage device. Alternatively, deterministic placement algorithms can be used to locate the correct storage devices based on the chunk's identifier

V. Security Analysis

The evaluation of the two secure deduplication models that we have presented is intended to demonstrate that the system is secure in the face of a variety of foreseeable scenarios. First, we examine the attacks that an external adversary could inflict upon the system. Second, we examine the security leaks possible when faced with a malicious insider who might have access to all of the raw data, such as system administrator with root-level access. Third, we examine the security implications involved when the keys in the system become compromised.

5.1 External Adversaries

For a system to be considered secure, it must be able to prevent information from leaking to an external attacker. A passive example of such an adversary would be an attacker that intercepts messages sent between players in the system. An active example is an adversary that changes or transmits messages. In both the authenticated and external model, the passive attacker problem is largely ameliorated by having the client perform the chunking and encryption. Thus, plaintext data is never transmitted in the clear. However, the anonymous model assumes that the keys can be exchanged in a secure manner but does not explicitly state how this is accomplished. A potential area of future work could be to define a secure protocol for this procedure. Since data transmitted between players is always encrypted, the danger from an active adversary is one of messages being changed. For example, in the basic models we have presented, a chunk could be intercepted en route to the chunk store and modified. While our design does not explicitly address such scenarios, these attacks can be largely mitigated through the use of transport layer security (TLS) approaches such as Secure Sockets Layer. As the anonymous model includes the goal of hiding the user's identity, an external adversary can gain some information by identifying where requests originate from. As with the man-in-themiddle type attacks previously discussed, our system does not directly deal with the issue, however solutions such as onion routing have addressed this concern, and are compatible with our design [12].

5.2 Internal Adversaries

As discussed in Section 3, a secure system must also provide protection from internal attackers. To this end, we analyze the ability of an inside adversary to launch attacks based on their location within the system and across their potential access levels. As in most systems, a malicious insider with full access can change or delete any information he chooses, resulting in a denial of service attack. From a security standpoint, our goal is, therefore, to limit an insider's ability to make targeted changes. There are two facets to limiting such changes. First, we would like to limit an insider's ability to target specific files. Second, we would like to limit an adversary's ability to make undetectable changes; overwriting a value with garbage is generally more detectable that overwriting it with a semantically valid, but incorrect value.

5.2.1 Authenticated Model

In the authenticated model, the metadata server does leak some information to an internal adversary. First, an insider has access to the file name to inode mapping. Second, the inode number to encrypted map entry is also available to an internal adversary. Finally, a malicious insider can determine the files to which a user has access, and the users that have access to a specific

file. Using the information available, a inside attacker at the metadata server is able to launch a variety of attacks. First, an inside adversary can delete metadata and revoke access for specific users. If the client is not knowledgeable about which files it should be able to access, this attack is undetectable. Second, when a client requests a file, the map entry of a different file accessible by the client could be returned. Whether or not this attack is detected would rely upon the client's understanding of the file's contents. Targeted changes to file contents, however, require the adversary to obtain the map key. In the current design, users grant access by submitting map keys encrypted using the authorized user's public key. In this way, a malicious insider is never exposed to the plaintext key needed to access a map entry's details. If the system were to encrypt map keys, a malicious insider could change the contents of map entries. One way to further strengthen the system, then, would be to hide the map entry from an inside attacker. This could be accomplished using a technique such as the anonymous model's map references, which, as shown in Figure 4, requires the map key in order to locate the map entry. Finally, if a malicious insider at the metadata store also distributes capability tickets, as is done in some systems, then it can be assumed that the adversary also has access to chunks; a malicious metadata store can simply issue itself a valid capability. However, without access to the map key, the adversary would not know which chunks correspond to a give file, and would lack the key needed to decrypt a chunk.

5.2.2 Anonymous Model

In both the authenticated and anonymous model, an inside adversary at the chunk store would be unable to modify data without being detected. Since the name of the chunk is based on the content, a user would not be able to request the modified chunk, or at the very least could tell that the chunk they requested is different from the chunk that was returned to them. An insider at the chunk store could, of course, delete chunks or refuse to fulfill chunk requests. In the anonymous model, the metadata store does leak some information to an internal adversary. First, an insider can deduce which inode numbers map to which files. This is not a serious issue because the user's symmetric key is needed to map inodes to map references. More importantly, however, an insider could deduce which entries are map references, as they will all be the same length. This is due to the fact that their payload is always one key, as opposed to a variable list of chunk metadata. One way to avoid leaking the fact that an entry is a map-key is to append some amount of random data to the entry.

5.3 Key Compromises

Any system that utilizes cryptographic primitives is highly dependent on the controlled access of encryption keys for the security of the system. As Kerckhoff's principle states, the security of the system comes from an adversary not knowing the encryption key; it is assumed that the adversary knows the protocols and cryptosystems. Thus, one way to analyze a security system is to examine the effects of compromised keys.

5.3.1 Authenticated Model

In the authenticated model, the user's identity is tied to their asymmetric key pair. Further, if an adversary learns a users private key, it is assumed they have the users complete key pair; the public key can easily be acquired from a certificate server. In this scenario, a malicious user may be able to fully impersonate the key's rightful owner, and obtain all the abilities of that user. As a safeguard against this possibility, it is recommended that authentication require more information than the user's key, but this approach is outside the scope of our model. A compromise of the other metadata key in the authenticated model, the map key, results in a less drastic information leak. If an adversary learns the map key, the problem of authenticating to the metadata store still exists. Finally, the revocation process can be used to generate a new map key, making the old key invalid. Thus, the system is relatively safe in the event of a compromised map key. Similarly, if the last key of the authenticated model, the chunk key, is compromised, the information leak is rather small. This is due to the fact that an adversary with the chunk key would still need to know the chunk identifier, and be able to authenticate to the chunk server in order to obtain plaintext data.

5.3.2 Anonymous Model

In the anonymous model, the user's private, symmetric key is very important to the security of the system. If a malicious user obtains the user's key, it can be safely assumed that they can access any file that the user has

stored a map reference for. Another potential attack they can issue in this scenario is to extend the length of the linked list of map entries indefinitely. However, since the anonymous model uses immutable chunks, a new key could be generated, and the file branched. If an adversary obtains the map key, the adversary will only need the inode number of the file to obtain plaintext data. Assuming that the number of inodes is relatively small, this can be accomplished using a brute force attack. Additionally, as the system is immutable, even generating a new map key will result in the original file being compromised. As in the authenticated model, an adversary with the chunk's encryption key, would still need to know the chunk identifier in order to obtain plaintext data.

VI. Proposed System

The basic idea in this paper is that we can exclude duplicate copies of storage data and limit the destruction/damage of stolen data if we can reduce the value of that stolen information to the attacker. This paper makes the first such attempt to properly address the problem of achieving efficient and reliable key management in secure deduplication. We recommend for providing security against insider attackers as well as outsider attackers and monitoring them by using Dekey, user behavior profiling and Decoy Technology. Dekey is a new construction in which users do not need to manage/store any keys on their personal but instead can securely distribute the convergent key shares through multiple servers. Dekey using the Ramp secret sharing scheme (RSSS) validates that Dekey incurs limited overhead in realistic environments. We propose a new structure called Dekey, which provides efficacy and reliability, guarantees for convergent key management on user, cloud storage and service provider sides. A new construction Dekey is proposed to provide effective and reliable convergent key management over convergent key Deduplication and secret sharing. Dekey supports both whole file-level and fixed/variable sizes block level Deduplication. Security analysis proves that Dekey is secure in terms of the definitions specified in the proposed security model. In particular, Dekey remains protected even if the adversary controls a limited number of key servers. We device Dekey using the Ramp secret sharing scheme(RSSS) that enables the key management to adapt to different reliability and confidentiality levels. Our assessment demonstrates that Dekey incurs limited procedures in normal upload/download operations in considerable cloud environments.

Fig. Proposed system Architecture

VII. 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 con-duct test bed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

FUTURE WORK

While the models we have presented demonstrate some of the ways that security and deduplication can coexist, works remains to create a fully realized, secure, space efficient storage system. Open areas for exploration exist in both security, as well as deduplication. Storage efficiency can be increased in a number of ways through intelligent chunking procedures. For example, the size of the file may be used to determine the average

chunk size, potentially yielding greater deduplication in data such as media files, which tend to be large and exhibit an “all or nothing” level of similarity with other files. However, since some large files, such as mail archives or tar files, may be aggregations of smaller files, another possibility would be to adjust chunking parameters based on file types. Since chunking is done at the clients rather than at the servers, this approach only requires that clients agree on the way they divide files into chunks. Moreover, taking this approach does not increase the likelihood of collision, which remains very small for chunk identifiers of 160 bits or longer. Unfortunately, techniques.

REFERENCES

[1] R. Di Pietro and A. Sorniotti. Boosting efficiency and security in proof of ownership for deduplication. In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security,ASIACCS ’12, pages 81–82, New York, NY, USA, 2012. ACM. [2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted deduplication. In Proc. of USENIX LISA, 2010.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. In USENIX Security Symposium, 2013.

[4] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In EUROCRYPT, pages 296– 312, 2013.

[5] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.

[6] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concur-rent attacks. In CRYPTO, pages 162–177, 2002.

[7] 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.

[8] 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.

[9] D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.

[10] R. C. Merkle. A digital signature based on a conventional encryption function. In A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, CRYPTO ’87, pages 369–378, London, UK, UK, 1988. Springer-Verlag.

Authors

Gangula Balaji pursuing M.Tech in Department of Computer Science and Engineering from D.N.R College of Engineering & Technology, Bhimavaram, Andhra Pradesh, West Godavari District, 534201, India. His area of interest in Cloud Computing, Services.

Dr. B.V.S Varma is working as Professor & Vice-Principal in the Department of Computer Science and Engineering in D.N.R College of Engineering & Technology, Bhimavaram, Andhra Pradesh, West Godavari District, 534201, India.

Dr. G. Satyanarayana is working as professor & HoD in the Department of Computer Science and Engineering in D.N.R College of Engineering & Technology, Bhimavaram, Andhra Pradesh, West Godavari District, 534201, India.