International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 7 (2015) pp. 17553-17561

http://www.ripublication.com

Secure Data Deduplication Using Efficient Cokm In Cloud Storage

Sreeram.G[1] and Arun.M[2]

[1]Assistant Professor,
Department Of Computer Science and Engineering
Dr. M.G.R.Educational & Research Institute University
Chennai-600095
[2]PG Scholar, Department Of Computer Science and Engineering
Dr. M.G.R.Educational & Research Institute University
Chennai-600095 Tel: +91 9884433666
sreeram@drmdgrdu.ac.in1 Tel: +91 9600652291, arasfr@gmail.com2

Abstract

Data deduplication is a technique which stores the data only once (no duplicate data can be stored in the cloud storage). Deduplication is used to reduce the storage space in the cloud and to efficiently use the bandwidth for uploading the data in cloud storage. Secure deduplication in cloud storage is one of the most auspicious and arising challenge. Although the convergent key encryption and Dekey are expansively implemented for secure deduplication. In convergent key management it is required to maintain the huge amount of convergent keys, when the number of users increases. Each user has to maintain a master key to protect the data. In Dekey, the large numbers of hash keys are generated. In order to overcome this problem a new Dekey which will generate only a fixed number of hash keys. If fixed amount of hash keys is generated the security for the hash keys are compact. In order to secure the hash keys an encryption standard was introduced. By using these standards the limits overhead in real time environment will be demonstrated.

Key terms: Dekey, dedupliaction, converget keys, hash keys.

Introduction

The initiation of cloud computing persuades the initiatives and Administrations outsource information to storage in third-party cloud suppliers, as demonstrated by much tangible case Studies [3]. One perilous challenge of today's cloud storage

services is the management of the aggregate volume of data. In 2020, it is expected that volume of the data will reach a peak of 40 trillion gigabytes, concerning report of IDC. Deduplication is a technique used to make data-management scalable and reduce the storage space, upload bandwidth in the cloud. A unique data copy is stored in the cloud and other repetitive data copies are referred to the unique data copy. Each such copy can be defined based on: fixed-size or variable-size data block (i.e., block-level deduplication). To reduce the maintenance cost in viable cloud storage such as Dropbox, Mozy and Memopal deduplication is the technique applied to the user data [12].

Data outsourcing increases confidentiality and privacy apprehension for a user. User relay on third-party to ensure that the cloud is properly secured from any attackers (i.e insider and outsider attacks). Using traditional encryption its unable to improve the storage and bandwidth efficiency. Each user is given a key for encryption and decryption of data in traditional encryption algorithms. Thus, unique cipher texts will be generated for each user (i.e different users), the same data copies cannot be identified using traditional algorithm which makes deduplication impossible.

Convergent encryption [8], ensure data privacy while comprehending deduplication. Convergent algorithm generates a convergent key with which the data's are encrypted and decrypted, convergent key generate a cryptographic hash value for the data copy itself [8]. The users cypher text that is generated by encryption of the data is transmitted to the cloud storage. When numbers of users needs to store identical data copy, the convergent encryption will generate same cipher text and convergent key. This allows the cloud service to eliminate the redundant cipher text. The cipher text that is generated by a user will be decrypted only by the particular user.

The convergent keys that have been introduced are stored in a sequential order along with the number of blocks and the number of users. The convergent key management becomes more noticeable in block-level deduplication. For example, suppose 1 TB of data is to be stored by a user, the data is divided into a unique block size of 4 KB each and SHA-256 algorithm generate a hash value for the data block, which is used in Drop box for deduplication [17]. The total size of the keys generated will be 8 GB. When the numbers of user increases the numbers of keys generated will be increased. This rigorous key management leads to colossal storage cost. The user has to pay for storing large numbers of keys in the cloud.

Dekey, which provides efficiency and reliability guarantees for convergent key management on both user and cloud storage sides. We propose an idea to implement deduplication to convergent keys and influence secret sharing technique. The convergent keys has been distributed across multiple independent key servers has been constructed. The secret share is distributed and computed for the first user who uploads the data mean while other users no need to store and compute the data. if the user has to recover the data copies, the user needs only the minimum number of key servers through authentication and attain the surreptitious shares to reconstruct the convergent keys. This means that the secret shares of the convergent keys will only be accessed by authorized users who own of the data copy. This significantly reduces the

storage overhead of the convergent keys and makes the key management reliable against failures and attacks. This paper makes the following contributions.

- A new construction is proposed to provide efficient and reliable convergent key management through convergent key deduplication and secret sharing.

Preliminaries

In this segment, we formally describe the cryptographic primitives used in secure deduplication.

Symmetric Encryption

To encrypt and decrypt the information the user uses a common secret key. There are three primitive functions.

- $KeyGen_{SE}(1^{\lambda})\rightarrow$using the security parameter 1^{λ}, K is the key generated using key generation algorithm K.
- $Encrypt_{SE}(k,m)\rightarrow C$ is the cipher text that is generated by the symmetric encryption algorithm C that takes the secret K and message M as input.
- $Decrypt_{SE}(k,c)\rightarrow M$ is the original message retrieved from the cipher text C and secret K using symmetric decryption algorithm M.

Convergent Encryption

Data confidentiality is provided by convergent encryption in deduplication [5],[8]. From the original data copy the user (data owner) derives a convergent key and the data block is encrypted using the key generated by the convergent encryption. A tag is derived for the data copy which is used to identify the duplicate data copy. The tag accuracy assets [5] clench, i.e., same tag will be generated if the data copies are same. The tag generated for the data copies are send to server side for the duplicate check that has been already stored. Both the rag and the convergent keys are autonomously derived. The convergent keys cannot be construed using the tag and involve the data concealment. The tag generated for the data individual copy and the encrypted data copy will be stored in the cloud. The convergent encryption scheme has four basic utilities:

- $KeyGen_{CE}(M)\rightarrow$the data copy M is mapped to a convergent key K by a key generation algorithm K;
- $Encrypt_{CE}(K,M) \rightarrow$ cipher test C is generated as a output, by using the data copy M and the convergent key as the input for the encryption algorithm (symmetric) C;
- $Decrypt_{CE}(K,) \rightarrow$ the original data copy M is derived as an output, by using the cipher text C and the convergent key K as the input for the decryption algorithm M ;
- $TagGen_{CE}(M)\rightarrow$ tag T(M) is the output generated from the original data copy M using a tag generation algorithm T(M). By using $T(M)=TagGen_{CE}(C)$, where $C=Encrypt_{CE}(K,M)$, tag is generated for the corresponding cipher text [5].

Proof of Ownership

A small hash value is used to solve the problem of proof of ownership (PoW) as a substitution for the entire file in the client side is used to avoid storing redundant files [11], where the antagonist could use the storage service as a content distribution network. PoW delivers a clarification to safeguard the security in client-side deduplication. In this way, a user can prove to server that is certainly has the file. Dekey along with PoW allows the client to prove their ownership of data copies to the storage server. Precisely, PoW is employed as an collaborating algorithm (represented by PoW) run by a user and by storage server. The storage server originates a small assessment ø(M) from a data copy M. To demonstrate the possession of the data copy M, the user needs to conduct and run a impervious algorithm with the storage server. It is approved if and only if ø = ø (M) and the proof is precise. In our paper , we use the representation of PoW_B to represent block B and PoW_F to represent file F. PoW protocol is represented by PoW_{Fj} with admiration to $Tj(F)=TagGen_{CE}(F, j)$.

Ramp Secret Sharing

RSSS (Ramp Secret Sharing scheme) is used to pile the convergent key by Dekey [6],[25]. Precisely, the (n,k,r)-RSSS (where 0=<r<k<n) engenders n segments from the furtive such that 1) the furtive can be convalesced from any k shares but cannot be convalesced from scarcer than K shares and 2) no facts about the furtive can be gathered from any r shares. It is known that when r=0, the (n,k,0)-RSSS becomes the (n,k) Rabin's Information Dispersal Algorithm (IDA) [23], when r=k-1, the (n,k,k-1)-RSSS becomes the (n,k) Shamir's Secret Sharing Scheme (SSSS) [26]. The (n,k,r)-RSSS forms on two embryonic utilities:

- Segment splits a secret S into (k-r) fragments of identical size, engenders r arbitrary fragments of identical size, and by using unorganized k-of-n elimination code[1] k fragments are encoded into n segments of identical size.
- *Recover* grosses slightly k out of n segments as inputs and then outputs the unique furtive S.

To make the engendered segments applicable for deduplication, we swap the overhead arbitrary fragments with pseudorandom fragments while implementing Dekey. RSSS is used by the Dekey to deliver tunable key administration to poise between privacy, consistency, storage overhead and routine.

Dekey

Convergent keys are maintained by the Dekey in an effective and dependable. It removes the redundant convergent keys and dispenses the convergent keys to multiple KM-CSPs. As a substitute for encrypting the convergent keys on a per-user basis, Dekey builds furtive segments on the unique convergent keys and dispenses the segments across various KM-CSPs. If a sequence of users segments the similar chunk, they can access the similar consistent convergent key. This expressively shrinks the storage overhead for the convergent keys. Dekey affords liability acceptance and allows the convergent key to endure reachable even if any subset of KM-CSPs miscarries.

Problem Formulations

System Model

There are three units for outsourcing the data by our model. They are the user, the cloud storage, key management cloud service provider.

- **User:** The user wants to uploads the data in the cloud storage and entrance the data whenever he poverties. In-order to condense the upload bandwidth and the storage interstellar the manipulator only uploads exclusive data. The data that are owned by the other user are not uploaded in the cloud.
- **Cloud storage:** It's used to store the user data and outsource the data stored by the user. The duplicate copies of data's are eliminated via deduplications and the cloud storage keeps only the unique data.
- **Key management cloud service provider:** It maintains the convergent keys. For liability forbearance of key management, we consider a minimum of KM-CSPs, each being an independent unit. Each convergent key is distributed across various KM-CSPs using RSSS.

In this work, we refer data copies to be a smaller block size, and this leads to deduplication. Block-level deduplication, which divides a file into smaller fixed-size and eliminates the duplicate data blocks to be stored in the cloud storage. Using fixed-size blocks shortens the reckonings of block boundaries. We organize our deduplication technique in block level. Specially, to upload a file, a user perform a block level verification and identifies the unique data block to be uploaded. Each data copy is associated with a tag for the duplicate check (see Section 2). All data copies and tags will be stored in the cloud storage.

Threat Model ad Security Goals

We deliberate two types of interlopers: 1) An external interlopers may attain certain familiarity of the data copy of curiosity via unrestricted network. It plays the role of the user that intermingles with the S-CSP. This kind of interlopers comprises the challenger who uses the S-CSP as a gratified dissemination network; 2) An private interlopers is authentic but inquisitive and it could refer to the S-CSP or any of the KM-CSPs. Its objective is to quotation useful facts of user data or convergent keys. We necessitate the private interlopers to follow the code of behavior correctly.

The consent between the S-CSP and KM-CSPs are allowed. The predefined inception r should be less than the number of conspired KM-CSPs if the (n, k, r)-RSSS is used (see Section 2), such that a convergent key cannot be predicted for an random memorandum by a brute-force attack from the conspired KM-CSPs. Our intention is to accomplish the following security objectives:

- Semantic security of convergent keys. The convergent keys that are stored in various KM-CSPs endure semantically protected, even if the challenger controls a predefined number of KMCSPs. These KM-CSPs are permissible to conspire with the S-CSP and the users. The goal of the challenger is to recover and retrieve the convergent keys for the records that do not be appropriate to them.

- Data concealment. We require the data copies that are encrypted should be semantically secure when they are volatile. Essentially, this prerequisite has recently been pompous in [4] and called the concealment against chosen distribution attack. This also includes the data privacy against the challenger who does not own the data. If the user do have the file then the particular user cannot get the ownership from the S-CSP and KM-CSPs by running the PoW protocol.

Implementation

In this session we will discuss implementation in detail. The Dekey is used to manage the convergent keys that are been generated and distribute the shares of convergent keys across multiple key servers by using ramp secret sharing.

Fig.1 presents the flow block diagram of the core module in this approach. In this figure, we have omitted the ordinary file transfer and duplication modules for simplification. To use the full use of the multi-core feature of the existing processors, we assume these modules running in parallel on different cores in a pipeline style. In Dekey we encode the hash key H0 with the other hash keys that are generated (H0,H1,H2…..Hn).

A fixed numbers of hash keys have been generated for the chunk data. We choose 12 kb as a default data block size. Larger the data block size results in better encoding/decoding, only fewer data chunks are to be managed [7],[16],[29]. Only a few numbers of hash keys have been used, a private key encryption is implemented in-order to provide security for the hash keys. A hash key of 32 bytes is generated using SHA-256 algorithm for each data block. SHA-256 belongs to the family of SHA-2 which is now recommended by the US National Institute of Standards and Technology (NIST) [2]. In addition, we adopt the symmetric-key encryption algorithm BLOWFISH in Cipher-Block Chaining (CBC) mode as the default encryption algorithm. Both SHA-256 and BLOWFISH are implemented.

We implement the RSSS based on Jerasure Version 1.2[20]. Regarding to the encoding, decoding, encryption and decryption in fig.1

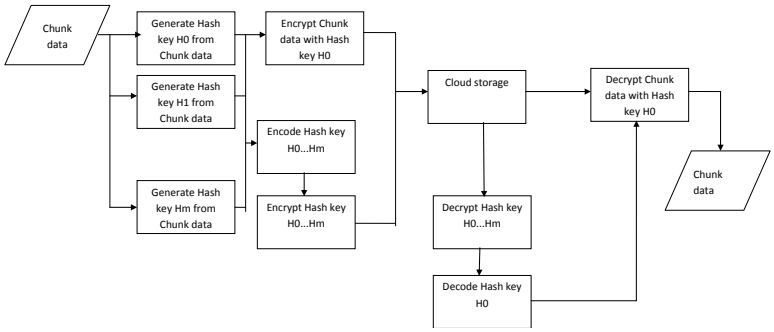

Figure 1:

Future Enhancement

Instead of using convergent encryption for the generation of hash keys various encryption can be used for the generation of the keys and the key length can be further increased in-order to get higher efficiency. Encoding of the hash keys can be removed and double encryptions can be used to protect the hash keys.

Conclusion

We have proposed a new Dekey approach in which each file is divided into number of blocks and they are verified for deduplication. In this approach convergent key duplication is eliminated efficiently and the data confidentiality of the outsourced data is maintained. We have implemented double encryption (i.e) one for the encoded hash keys, in-order to maintain the confidentiality the keys and the data encryption is the second one in which blowfish encryption is implemented for the outsourced data. The new Dekey use RSSS for the encryption/decryption of the encoded hash keys with the regular upload and download operations.

References

[1] NIST's Policy on Hash Functions, Sept. 2012. [Online]. Available:http://csrc.nist.gov/groups/ST/hash/policy.html.

[2] P. Anderson and L. Zhang, ''Fast and Secure Laptop Backupswith Encrypted De-Duplication,'' in Proc. USENIX LISA, 2010,pp. 1-8.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart, ''Message-LockedEncryption and Secure Deduplication,'' in Proc. IACR CryptologyePrint Archive, 2012, pp. 296-3122012:631.

[4] G.R. Blakley and C. Meadows, ''Security of Ramp Schemes,'' inProc. Adv. CRYPTO, vol. 196, Lecture Notes in Computer Science,G.R. Blakley and D. Chaum, Eds., 1985, pp. 242-268.

[5] A.T. Clements, I. Ahmad, M. Vilayannur, and J. Li, ''Decentralized Deduplication in San Cluster File Systems,'' in Proc.USENIX ATC, 2009, p. 8.

[6] J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer,''Reclaiming Space from Duplicate Files in a Serverless DistributedFile System,'' in Proc. ICDCS, 2002, pp. 617-624.

[7] J. Gantz and D. Reinsel, The Digital Universe in 2020: Big Data,Bigger Digital Shadows, Biggest Growth in the Far East, Dec. 2012[Online]. Available:http://www.emc.com/collateral/analystreports/idc-the-digital-universe-in-2020.pdf.

[8] R. Geambasu, T. Kohno, A. Levy, and H.M. Levy, ''Vanish:Increasing Data Privacy with Self-Destructing Data,'' in ProcUSENIX Security Symp., Aug. 2009, pp. 316-299.

[9] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg,''Proofs of Ownership in Remote Storage Systems,'' in Proc.ACM Conf.

Comput.Commun. Security, Y. Chen, G. Danezis,and V. Shmatikov, Eds., 2011, pp. 491-500.

[10] D. Harnik, B. Pinkas, and A. Shulman-Peleg, ‘‘Side Channels inCloud Services: Deduplication in Cloud Storage,’’ IEEE SecurityPrivacy, vol. 8, no. 6, pp. 40-47, Nov./Dec. 2010.

[11] S. Kamara and K. Lauter, ‘‘Cryptographic Cloud Storage,’’ inProc. Financial Cryptography: Workshop Real-Life Cryptograph.Protocols Standardization, 2010, pp. 136-149.

[12] M. Li, ‘‘On the Confidentiality of Information Dispersal Algorithmsand their Erasure Codes,’’ in Proc. CoRR, 2012, pp. 1-4abs/1206.4123.

[13] D.T. Meyer and W.J. Bolosky, ‘‘A Study of Practical Deduplication,’’in Proc. 9th USENIX Conf. FAST, 2011, pp. 1-13.

[14] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, andE. Weippl, ‘‘Dark Clouds on the Horizon: Using Cloud Storageas Attack Vector and Online Slack Space,’’ in Proc. USENIXSecurity, 2011, p. 5.

[15] W.K. Ng, Y. Wen, and H. Zhu, ‘‘Private Data DeduplicationProtocols in Cloud Storage,’’ in Proc. 27th Annu. ACM Symp.Appl. Comput., S. Ossowski and P. Lecca, Eds., 2012,pp. 441-446.

[16] R.D. Pietro and A. Sorniotti, ‘‘Boosting Efficiency and Security in Proof of Ownership for Deduplication,’’ in Proc. ACM Symp. Inf., Comput. Commun. Security, H.Y. Youm and Y. Won, Eds., 2012pp. 81-82.

[17] J.S. Plank, S. Simmerman, and C.D. Schuman, ‘‘Jerasure: ALibrary in C/C++ Facilitating Erasure Coding for StorageApplicationsVVersion 1.2,’’ University of Tennessee, Knoxville,TN, USA, Tech. Rep. CS-08-627, Aug. 2008.

[18] M.O. Rabin, ‘‘Fingerprinting by Random Polynomials,’’ Centerfor Research in Computing Technology, Harvard University, Cambridge, MA, USA, Tech. Rep. TR-CSE-03-01, 1981.

[19] A. Rahumed, H.C.H. Chen, Y. Tang, P.P.C. Lee, and J.C.S. Lui,‘‘A secure Cloud Backup System with Assured Deletion and Version Control,’’ in Proc. 3rd Int’l Workshop Security Cloud Comput., 2011, pp. 160-167.

[20] A. Shamir, ‘‘How to Share a Secret,’’ Commun.ACM, vol. 22, no. 11, pp. 612-613, 1979.

[21] M.W. Storer, K. Greenan, D.D.E. Long, and E.L. Miller, ‘‘Secure Data Deduplication,’’ in Proc. Storage S S, 2008, pp. 1-10.

[22] Y. Tang, P.P. Lee, J.C. Lui, and R. Perlman, ‘‘Secure Overlay Cloud Storage with Access Control and Assured Deletion,’’ IEEE Trans. Dependable Secure Comput., vol. 9, no. 6, pp. 903-916, Nov./Dec. 2012.

[23] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, ‘‘Characteristics of Backup Workloads in Production Systems,’’ in Proc. 10th USENIX Conf. FAST, 2012, pp. 1-16.

[24] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, ‘‘Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847-859, May 2011.

[25] W. Wang, Z. Li, R. Owens, and B. Bhargava, ‘‘Secure and Efficient Access to Outsourced Data,’’ in Proc. ACM CCSW, Nov. 2009, pp. 55-66.
[26] Z. Wilcox-O’Hearn and B. Warner, ‘‘Tahoe: The Least-Authority Filesystem,’’ in Proc. ACM StorageSS, 2008, pp. 21-26.
[27] A. Yun, C. Shi, and Y. Kim, ‘‘On Protecting Integrity and Confidentiality of Cryptographic File System for Outsourced Storage,’’ in Proc. ACM CCSW, Nov. 2009, pp. 67-76.