International Journal of Research

International Journal of Research (IJR)

e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 09, September 2015

Available at http://internationaljournalofresearch.org

Efficient and Reliable Convergent Key Management with Secure Deduplication

Rawula Rohith Raj

M.Tech

S.Ravi Kumar

Ph.D. Assistant Professor

ANURAG Group of Institutions (CVSR College of engineering)

Abstract:

Data deduplication is a technique for eradicate reproduction copy of data and has been extensively used in cloud storage to reduce storage liberty and upload bandwidth. Hopeful as it is a happen challenge is to perform protected deduplication in cloud storage space. Even though convergent encryption for comprehensively implement for secure deduplication a dangerous problem of making convergent encryption realistic is to professionally and dependably handle a huge number of convergent keys. This paper makes the first effort to formally address the problem of accomplish efficient and dependable key management in secure reduplicate. We first establish a baseline come within reach of user clutch an independent master key for encrypting the convergent keys and outsourcing them to the cloud. However such a baseline key organization scheme generates a huge number of keys with the growing number of users and necessitates users to dedicatedly shelter the keys. We proposition dekey a new production in which users do not need to administer any keys on their own but as an alternative securely allocate the convergent key contribute to across several servers. Security investigations demonstrate that Dekey is secure in terms of the description specified in the projected precautions. A verification of concept we implement Dekey using the secret sharing scheme and display that Dekey deserves limited overhead in realistic location.

Keywords: Data DE duplication; upload bandwidth; key management; scheme generates; Dekey Secure

Introduction: Cloud storage offers highly-available virtually infinite and quick to scale storage with its pay as you go model in recent years it has attracted new customers by the score. Coupled with dropping prices the cloud paradigm has turned storage into a commodity. The decreasing cost of storage media the use of multi tenancy competition between cloud providers and the efficient use of the storage backend through compression and deduplication can be listed amongst the reasons for low price high quality cloud services such as cloud storage services. One of the techniques used to reduce the cost of cloud storage services is deduplication which is currently implemented by providers such as deduplication avoids storing multiple copies of the equivalent data. As an illustration multiple copies of popular content need to be stored only once upon the upload subsequent upload requests can be discarded and only require establishing a link from the uploading user to the original copy of the content. Deduplication can be performed very effectively at both or block level deduplication ratios vary from 2:1 to 50:1 for the same application by the same vendor depending on the setup and the input dataset. Deduplication can take place at the client side or at the server side. If deduplication is triggered at the client side it is more efficient as it saves upload bandwidth. This is especially for service providers due to the fact that network activity is the most energy consuming task for cloud. To keep away from the transmission of the complete content but still allowing to check for its existence at the server side clients are usually asked to generate a much shorter version of and to use that digest to uniquely identify. The standard approach is to interpret the upload of a digest by a client as a proof that the client essentially owns that. In the work patterned the protection weaknesses hidden behind approach. Primary the privacy and confide gentility of users of a storage system can be compromised by an attacker that checks if another user has already uploaded a by trying to upload it as well. If the upload does not take place it means the server already stores it. This can be extremely dangerous if the is very rare or private. Second deduplication can be abused to turn the service provider into an underground direct. Two join together users with no direct connectivity can establish a protocol to exchange information stealthily. For instance to exchange one bit of information one of the users checks if a previously agreed has been uploaded or not during a certain time window If the was uploaded the user can consider that a 1 has been transmitted. Finally a cloud storage service can be used as a content distribution network (CDN). In such a case a user can share large with other users just by exchanging the consequent assimilate. A real world illustration of this attack was the explosive growth of digital contents continues to raise the demand for new storage and network capacities along with an increasing need for more cost effective use of storage and network bandwidth for data transfer.

Overview: The advent of cloud storage motivates enterprises and organizations to outsource data storage to third party cloud providers as evidenced by many real life case studies [3]. One critical challenge of today's cloud storage services is the management of the ever increasing the data. According to the investigation report of IDC of the data in the wild is expected to reach in [9]. To make data management scalable deduplication has been a well known technique to reduce storage space and upload bandwidth stored in cloud. Its position of observation more than data copies with the same content deduplication eliminates redundant data by keeping only one physical copy and referring other redundant copy of data. Every such copy can be definite based on different granularities it may refer to either a complete organizer. Today's profitable storage services in the cloud such as have been applying deduplication to user data to save maintenance cost [12]. From a user's perspective data outsourcing raises security and isolation anxiety. We must belief third party cloud contributor to properly enforce confidentiality integrity checking and access control mechanisms against any insider and unknown attack from the storage data. However deduplication while improving storage and bandwidth efficiency is incompatible with conservative encryption. Particularly established encryption involve different users to encrypt their data with the own keys. Consequently matching data copies of different users will lead to different cipher texts making reduplication impracticable. Convergent encryption [8] provides a viable option to enforce data confidentiality while realizing reduplication. It encrypts or decrypts a data copy with a convergent key which is derived by computing the cryptographic hash value of the content of the data copy itself [8]. After key generation and data encryption keys and sends the cipher text to the cloud. The encryption is deterministic matching data copies will generate the same convergent key and the same cipher text. This allows the cloud to perform deduplication on the cipher texts. In this texts can only be decrypted by the corresponding data owners of the particular keys. To understand how convergent encryption can be understand we consider a baseline approach that implements convergent encryption based on a together with this approach. That is the original data copy is first encrypted with a convergent key derived by the data copy itself and the convergent key is then encrypted by a master key that will be kept locally and securely by every user. The encrypted convergent keys stored beside with the corresponding encrypted data copies in stored in the cloud. The key can be used to recuperate the encrypted keys and hence the encrypted files. Each user only needs to continue the master key and the metadata about the sourced data. Conversely the baseline approach suffers two critical consumption issues. It is inefficient as it will produce an enormous number of keys with the increasing quantity of users. Particularly every user must connect an encrypted convergent key with every block of its outsourced encrypted data copies so as to later reinstate the information. Although unusual users may contribute to the matching data copies they must have their own set of convergent keys so that no other users can admission files. As a consequence the quantity of convergent keys being introduced linearly scales with the number of blocks being stored and the quantity of users. The key organization overhead becomes more prominent if we exploit fine grained block level deduplication. Suppose that a user stores 1 TB of data with all unique blocks of size 4 KB each and that each convergent key is the hash value of SHA-256 which is used by reduplication [17]. The quantity of keys is further multiplied by the number of users. The resulting intensive key management overhead leads to the huge storage cost as users must be billed for storing the large number of keys in the cloud. The baseline approach is unreliable as it requires each user to dedicatedly protect own key. If the key is accidentally lost then the user data cannot be recovered if it is compromised by attackers then the user data leaked. These stimulate us to explore how to efficiently and reliably manage enormous convergent keys while still achieving secure deduplication. We propose a new manufacture called Dekey which provides efficiency and reliability guarantees for convergent key an argument on both user and cloud storage sides. In particular we construct secret shares for the convergent keys and distribute them across multiple independent key servers. The first user who uploads the data is required to compute and distribute such secret shares while all following users who own the same data copy need not compute and stored. To recover data copies a user must access a minimum number of key servers through authentication and obtain the secret shares to reconstruct the keys. The secret allocate of a convergent key will only be accessible by the authorized users who own the corresponding the copy of data. Considerably reduce the storage overhead of the convergent keys and makes the key management reliable against malfunction and attacks. To our information nothing of existing studies formally addresses the problem of convergent key management.

System Implementation: After careful analysis the system has been identified to have the following modules:

1. Secure Deduplication

2. User Behavior Profiling

3 .Decoy documents

Secure Deduplication: Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Related and somewhat synonymous terms are intelligent (data) compression and single-instance (data) storage. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced. This type of deduplication is different from that performed by standard file-compression tools, such as LZ77 and LZ78. Whereas these tools identify short repeated substrings inside individual files, the intent of storage based data deduplication is to inspect large volumes of data and identify large sections – such as entire files or large sections of files – that are identical, in order to store only one copy of it. This copy may be additionally compressed by single-file compression techniques. For example a typical email system might contain 100 instances of the same 1 MB (megabyte) file attachment. Each time the email platform is backed up, all 100 instances of the attachment are saved, requiring 100 MB storage space.

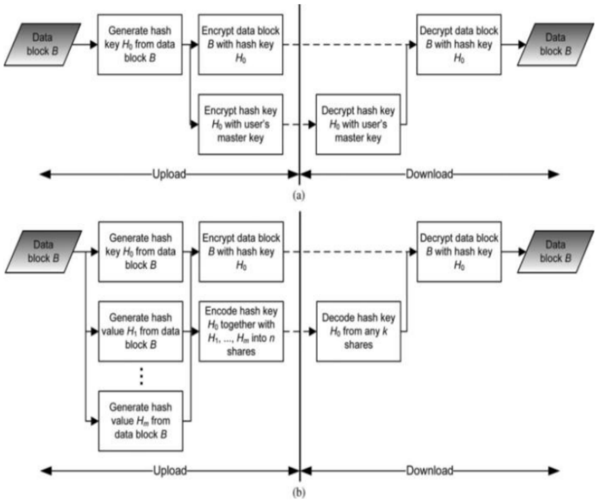
Data block B
Generate hash key H_0 from data block B
Encrypt data block B with hash key H_0
Decrypt data block B with hash key H_0
Data block B
Encrypt hash key H_0 with user's master key
Decrypt hash key H_0 with user's master key
Upload
Download
(a)
Data block B
Generate hash key H_0 from data block B
Encrypt data block B with hash key H_0
Decrypt data block B with hash key H_0
Data block B
Generate hash value H_1 from data block B
Encode hash key H_0 together with H_1, ..., H_m into n shares
Decode hash key H_0 from any k shares
Generate hash value H_m from data block B
Upload
Download
(b)

Fig 1: Secure deduplication
(a)Flow diagram keeping hash key
(b) Flow diagram of Dekey keeping hash key with RSSS.

User Behavior Profiling: Whereas these tools identify short repeated substrings inside individual files, the intent of storage based data deduplication is to inspect large volumes of data and identify large sections – such as entire files or large sections of files – that are identical, in order to store only one copy of it. This copy may be additionally compressed by single-file compression techniques. For example a typical email system might contain 100 instances of the same 1 MB (megabyte) file attachment. Each time the email platform is backed up, all 100 instances of the attachment are saved, requiring 100 MB storage space.

Advantages: 1. A new construction in which users do not need to manage any keys on their own

2. Instead securely distribute the convergent key shares across multiple servers

3. Security analysis demonstrates that Dekey is secure in terms of the definitions specified

4. Convergent encryption also known as content hash keying is a cryptosystem that produces identical cipher text

5. Identical plaintext files this has applications in cloud computing to remove duplicate files from storage without the provider having access to the encryption keys

Conclusion: The basic idea is that we posit that secure deduplication services can be implemented given additional security features insider attacker on Deduplication and outsider attacker by using the detection of masquerade activity. The confusion of the attacker and the additional costs incurred to distinguish real from bogus information, and the deterrence effect which, although hard to measure, plays a significant role in preventing masquerade activity by risk averse attackers. We posit that the combination of these security features will provide unprecedented levels of security for the deduplication.

Reference:

[1.] 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.

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

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

[4.] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, ‘‘Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space,’’ in Proc. USENIX Security, 2011, p. 5.

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

[6.] 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.

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

[8.] S. Kamara and K. Lauter, ‘‘Cryptographic Cloud Storage,’’ in Proc. Financial Cryptography: Workshop RealLife Cryptograph. Protocols Standardization, 2010, pp. 136-149.