A SURVEY ON KEY MANGEMENT SYSTEM ALONG WITH DEDUPLICATION TECHNIQUE

P. Balasubhramanyam Reddy
PG Scholar, Department of CSE,
Saveetha Engineering College,
Thandalam, Chennai, India
balredz@gmail.com

G. Nagappan
Associate Professor, Department of CSE,
Saveetha Engineering College,
Thandalam, Chennai, India
nagappan.cse@saveetha.ac.in

***Abstract*—Data deduplication is a method for removing duplicate copies of data, It has been largely used in cloud storage to reduce storage memory and upload bandwidth. It gives a challenge to do secure deduplication in cloud storage. In encryption methods the keys can be produced but cannot manage huge number of keys. In the first attempt to formally address the problem of achieving efficient and reliable key management in secure deduplication. The general approach in which each user holds an independent master key for encrypting the convergent keys and employing them to the cloud. such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to allegiance to protect the master keys. The De-key is the process ,which creates new construction in which users do not need to manage any keys on their own but instead of it secure distribute of the convergent key shares across multiple servers. Security analysis demonstrates that De-key is secure in the proposed security model. Proof is that in realistic environment the De-key used in ramp secret sharing .which can Demonstrate.**

***Keywords*—De-duplication, convergent encryption, key management, auditing.**

I INTRODUCTION

The advantage of cloud storage motivates enterprises and organizations to outsource data storage to third-party cloud providers. One critical challenge of today's cloud storage services is the management of the increasing volume of data. According to the report of IDC, the volume of data in the will expected to reach 50-60 trillion giga bytes in 2020. To make data management scalable, de-duplication has been a well-known technique to reduce storage space and upload bandwidth in cloud storage. Instead of keeping multiple data copies with the same content duplication redundant data by keeping only one physical copy and referring other redundant data to that copy. Each such copy can be defined based on different granularities: it may refer to either a whole file, or amore fine-grained fixed-size or variable-size. The commercial cloud storage services, such as Drop box, Mazy and Memo pal, have been applying deduplication to user data to save maintenance cost ,from the user side , data from outside may have doubt in security and privacy concerns. In this trust third-party cloud providers to properly enforce confidentiality, integrity checking, and access control mechanisms against any insider and outsider attacks. The de-duplication is improving storage and bandwidth efficiency, is incompatible with traditional encryption. Specially different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different cipher texts, making de-duplication impossible Convergent encryption provides a viable option to enforce data confidentiality while realizing de-duplication. It encrypts/decrypts data copy with a convergent key, which is derived by computing the cryptographic hash value of the content of the data copy itself . After key generation and data encryption, users retain the keys and send the cipher text to the cloud.

Due to encryption is deterministic, the same data which already exists copies will generate the same convergent key and the same cipher text. This allows the cloud to perform de-duplication on the ciphertexts. The ciphertexts can only be decrypted by the corresponding data owners with their convergent keys.

In baseline is approach suffers two critical deployment issues. First, it is inefficient, as it will generate an enormous number of keys with the increasing number of users. each user must associate an encrypted convergent key with each block of its outsource decrypted data copies, so as to later restore the data copies. Although different users may share the same data copies, they must have their own set of convergent keys so that no other users can access their files. As a result, the number of convergent keys being introduced linearly scales with the number of blocks being stored and the number of users. This key management overhead becomes more prominent if we exploit fine-grained block-level de-duplication.

Second, the baseline approach is unreliable, as it requires each user to dedicatedly protect his own master key. If the master key is accidentally lost, then the user data cannot be recovered; if it is compromised by attackers, then the user data will be leaked. us to explore how to efficiently and reliably manage enormous convergent keys, while still achieving secure de-duplication. To this end, we propose a new construction called De-key, which provides efficiency and reliability guarantees for convergent key management on both user and cloud storage sides.

II RELATED WORK

A. Traditional Encryption:

To protect the confidentiality of outsourced data, various cryptographic solutions have been proposed in the literature. The idea is to builds untraditional encryption, in which each user encrypts data with an independent secret key. Some studies which is used to propose the use of threshold secret sharing to maintain the robustness of key management.

These do not consider deduplication. Using traditional encryption, different users will simply encrypt identical data copies with their own keys, but this will lead to different cipher texts and hence make de-duplication impossible.

B. Convergent Encryption:

Convergent encryption ensures data privacy in de-duplication Bellaire Formalize this primitive as message-locked encryption, and explores its application in space-efficient secure outsourced storage. There are also several implementations of convergent implementations of different convergent encryption variants for secure de-duplication. It is known that some commercial cloud storage providers, such as Betas, also deploy convergent encryption . However, as stated before, convergent encryption leads to a significant number of convergent keys.

C. Proof of Ownership:

Halevietal. propose ‘‘proofs of ownership’’ (POW) ford duplication systems, such that a client can efficiently prove to the cloud storage server that he/she owns a file without uploading the file itself. Several POW constructions based on the Merle Hash Tree are proposed to enable client-side de-duplication, which include the bounded leakage setting. Pietro and Sorniotti propose another efficient POW scheme by choosing the projection of a file onto some randomly selected bit-positions as the file proof. Note that all the above schemes do not consider data

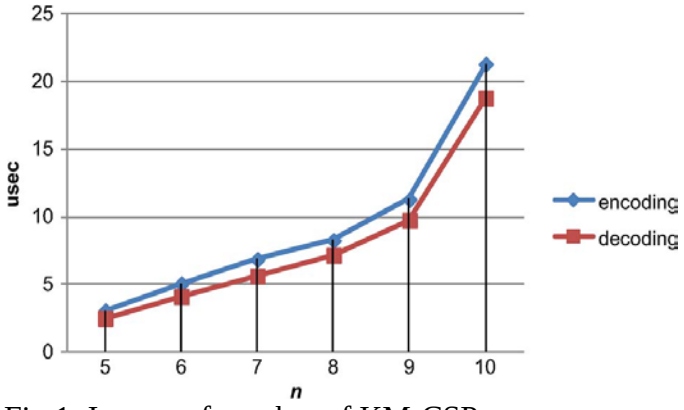

Fig 1: Impact of number of KM-CSPs n on encoding/decoding times, where r = 2 and n - k =2.

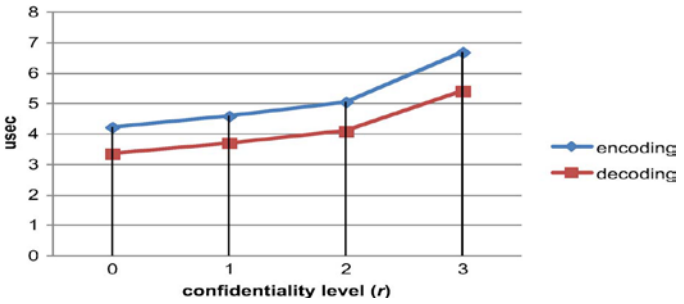

Fig 2:Impact of confidentiality level r on the encoding/decoding times where n=6

III ARCHITECTURE

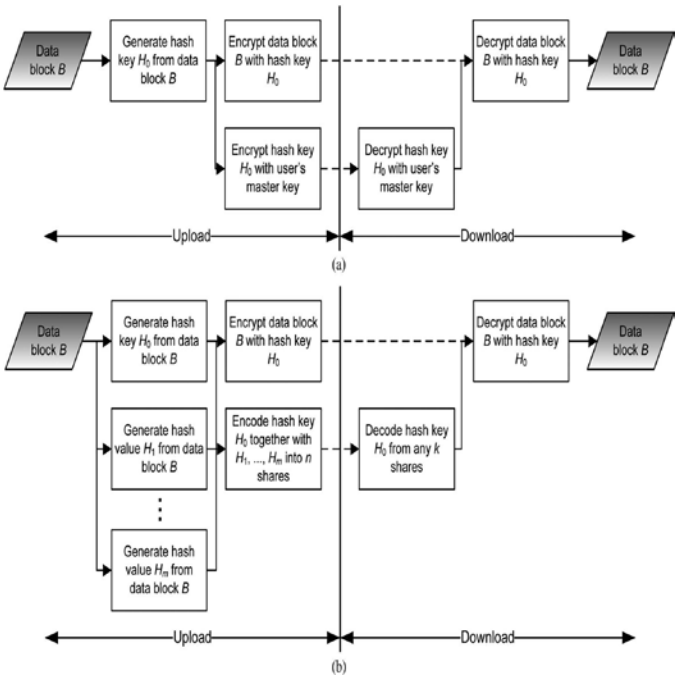

Fig 3:low block diagrams of core modules in two different approaches. (a) Baseline approach (keeping the hash key with an encryption scheme).(b) De-key (keeping the hash key with (n; k, r -RSSS).

Fig. 3 presents the flow block diagrams of core modules in the baseline approach and De-key that we implement. In this figure, we omit the ordinary file transfer and de-duplication modules for simplification. To make full use of the multi-core feature of contemporary processors, we assume that these modules running in parallel on different cores in a pipeline style. In the baseline approach, we simply encrypt each hash key H0 with the user's master key, while in De-key, we generate n shares of H0.We choose 4 KB as the default data block size. A larger data block size (e.g., 8 KB instead of 4 KB) results in better encoding/decoding performance due to fewer chunks being managed, but has less storage reduction offered by de-duplication. Which each data block, abash key of size 32 bytes is generated using the hash function SHA-256, which belongs to the family of SHA-2that is now recommended by the US National Institute of Standards and Technology (NIST). In addition, we adopt the symmetric-key encryption algorithm AES-256in Cipher-Block Chaining (CBC) mode as the default encryption algorithm. Both SHA-256 and AES-256 are implemented using the EVP library of OpenSSL Version1.0.1e.

We implement the RSSS based on Jerasure .Regarding to the encoding and decoding modules in Fig. 1b, the choice of code symbol size w (in bits) deserves our discussion here. For an erasure code, a code symbol of size w bits refers to a basic unit of encoding and decoding operations, both of which are performed in a finite field . In the RSSS, we choose the erasure code whose generator matrix is a Cauchy matrix, and thus, w should meet the condition. However, when each hash key is divided into pieces with a size of multiple w, its size (i.e., 32 bytes) is often not a multiple of w . We thus often need to pad additional zeros to fill in the Pieces, resulting in different storage
blow up ratios.

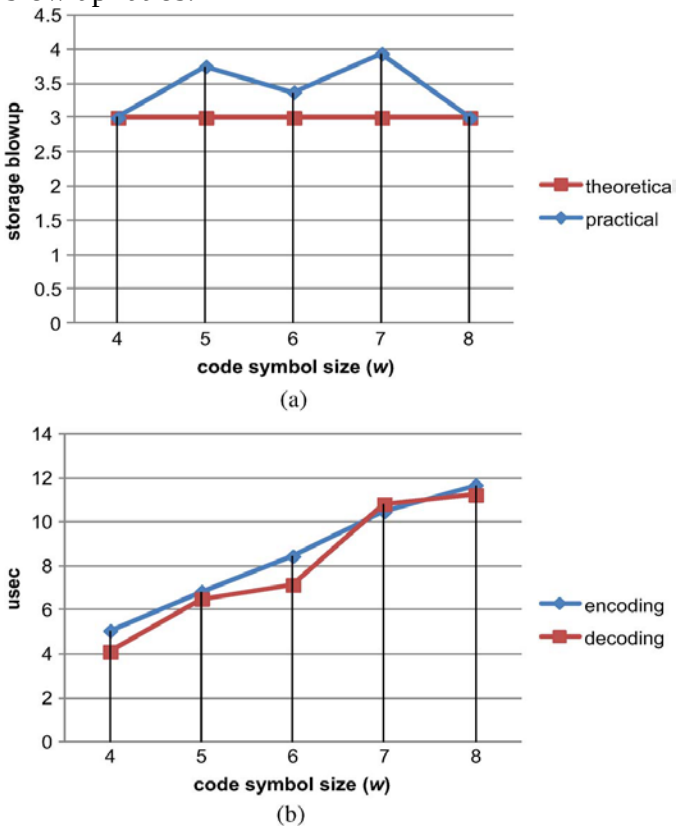

Fig. 2a shows the storage blowups ratios versus different values of w for (6, 4, 2)-RSSS. We see that for some w, the storage blowups ratio can be much higher than the theoretical value calculated by n. However, we find that if the minimum w is chosen, the practical storage blowup can often be closely matched to the theoretical value. In addition, we evaluate the corresponding encoding and decoding times on an Intel Xeon E5530 (2.40 GHz)server with Linux 3.2.0-23-generic OS, and the results are

shown in Fig. 2b. We find that the encoding and decoding times increase with w. Therefore, our De-key implementation always chooses the minimum w that meets w.

IV IMPLEMENTATION

In discuss of implementation details of De key. De-key builds on the Ramp secret sharing scheme(RSSS) to distribute the shares of convergent keys across multiple key servers.

A. RSSS with Pseudo randomness

In De-key, the RSSS secret is the hash key H0 of a data block B, where H0=hash(B) .Recall the Share function of the (n; k; r)-RSSS embeds r random pieces to achieve a confidentiality level of r. One challenges that randomization conflicts with de-duplication, since the random pieces cannot be de-duplicated with each other. Instead of directly adopting RSSS, we here replace these random pieces with pseudorandom pieces in our De-key implementation.

It generate the r pseudorandom pieces as follows. Let M=[r/(k-r)].The first generating m additional hash values asH1=hash(B+1);H2= hash(B+2); . . .;Hm=hash(B+ m).We then fill in the r pieces with the generated m additional hash values H1;H2; . . .;Hm. These r pieces are pseudorandom because

1. H1;H2; . . .;Hm cannot be guessed by attackers along as the corresponding data block B is unknown; and
2. H1;H2; . . .;Hm together with H0 cannot be deduced

from each other as long as the corresponding data block B is unknown.

The parameters n, k, and r determine the following four factors,

- Confidentiality level: It is decided by the parameter r.
- Reliability level: It depends on the parameters n and
  k, and can be defined by n _ k.
- Storage blow-up: It determines the key management
  overhead and depends on the parameters n, k, and r.
  It can be theoretically calculated by n /k-r.
- Performance: It refers to the encoding performance

and decoding performance when using the k-of-n erasure code in the Share and Recover functions, respectively.

Fig. 1 presents the flow block diagrams of core modules in the baseline approach and De-key that we implement. In this figure, we omit the ordinary file transfer and de-duplication modules for simplification. To make full use of the multi-core feature of contemporary processors, we assume that these modules running in parallel on different cores in a pipeline style. In the baseline approach, we simply encrypt each hash key H0 with the user's master-key, while in De-key, we generate n shares of H0.

The 4 KB is chosen as the default data block size. A larger data block size results in better encoding/decoding performance due to fewer chunks being managed, but has less storage reduction offered by de-duplication. For each data block, abash key of size 32 bytes is generated using the hash Function SHA-256, which belongs to the family of SHA-2that is now recommended by the US National Institute of Standards and Technology (NIST) . In addition, we adopt the symmetric-key encryption algorithm AES-256in Cipher-Block Chaining (CBC) mode as the default encryption algorithm. Both SHA-256 and AES-256 are implemented using the EVP library of Opens' Version1.0.10.

The implementation of RSSS based on Jerasure Version 1.2. Regarding to the encoding and decoding modules in Fig. 1b, the choice of code symbol size w (in bits) deserves our discussion here. For an erasure code, a code symbol of size w bits refers to a basic unit of encoding and decoding operations, both of which are performed in a finite field GF(2w). In the (n, k, r)-RSSS, we choose the erasure code .Theshould meet the condition $2w > n+k$. However, when each hash key is divided into (k- r) pieces with a size of multiple w, its size (i.e., 32 bytes) is often not a multiple of w multiplied with (k-r) we thus often need to pad additional zeros to fill in the (k-r) pieces, resulting in different storage blow up ratios.

V CONCLUSION& FUTURE WORK

TheDe-key is an efficient and reliable convergent key management scheme for secure de-duplication. De-key applies de-duplication among convergent keys and distributes convergent key shares across multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. We implement

De-key using the Ramp secret sharing scheme and demonstrate that it incurs small encoding/decoding overhead compared to the network transmission overhead in the regular upload/download operations.

The audit of the file sharing and time can be recorded and space can be utilise in various methods and make it less expensive de-duplication can also be tried in data warehousing although backup ,replication there yet to we can implement this technology we can help to make more free space and make It a low cost.

.

REFERENCES

1. A.Shamir, ‘‘How to Share a Secret,’’. ACM, vol. 22,no. 11, pp. 612-613, 1979.
2. M.W. Storer, K. Greenan, D.D.E. Long, and E.L. Miller, ‘‘Secure Data De-duplication,’’ in Proc. Storages, 2008, pp. 1-10.
3. 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 Computer., vol. 9, no. 6, pp. 903-916,Nov./Dec. 2012.
4. G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone,M.hamness, and W. Hsu, ‘‘Characteristics of Backup Workloads in Production Systems,’’ in Proc. 10th USENIX Conf. FAST,2012, pp. 1-16.
5. Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, ‘‘Enabling PublicAuditability and Data Dynamics for Storage Security in Cloud Computing,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5,pp. 847-859, May 2011.
6. 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.
7. Z. Wilcox-O’Hearn and B. Warner, ‘‘Tahoe: The Least-AuthorityFilesystem,’’ in Proc. ACM StorageSS, 2008, pp. 21-26
8. 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.
9. G.R. Blakley and C. Meadows, ‘‘Security of Ramp Schemes,’’ inProc. Adv. CRYPTO, vol. 196, Lecture Notes in Computer ScienceG.R. Blakley and D. Chaum, Eds., 1985, pp. 242-268.
10. A.T. Clements, I. Ahmad, M. Vilayannur, and J. Li, ‘‘DecentralizedDeduplication in San Cluster File Systems,’’ in Proc.USENIX ATC, 2009, p. 8.
11. J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer,‘‘Reclaiming Space from Duplicate Files in a ServerlessDistributed.File System,’’ in Proc. ICDCS, 2002, pp. 617-624.
12. J. Gantz and D. Reinsel, The Digital Universe in 2020: Big Data,Bigger Digital Shadows, Biggest Growth in the Far East, Dec. 2012.
13. R. Geambasu, T. Kohno, A. Levy, and H.M. Levy, ‘‘Vanish:Increasing Data Privacy with Self-Destructing Data,’’ in Proc.`USENIX Security Symp., Aug. 2009, pp. 316-299.
14. 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.
15. D. Harnik, B. Pinkas, and A. Shulman-Peleg, ‘‘Side Channels in Cloud Services: De-duplication in Cloud Storage,’’ IEEE SecurityPrivacy, vol. 8, no. 6, pp. 40-47, Nov./Dec. 2010.
16. S. Kamara and K. Lauter, ‘‘Cryptographic Cloud Storage,’’ inProc. Financial Cryptography: Workshop Real-Life Cryptograph.Protocols Standardization, 2010, pp. 136-149.
17. M. Li, ‘‘On the Confidentiality of Information Dispersal Algorithmsand their Erasure Codes,’’ in Proc. CoRR, 2012, pp. 1-4abs/1206.4123.
18. D. Meister and A. Brinkmann, ‘‘Multi-Level Comparison of DataDeduplication in a Backup Scenario,’’ in Proc. SYSTOR, 2009,pp. 1-12