MULTILEVEL CONVERGENT KEY MANAGEMENT FOR CLOUD STORAGE

[1]Nithiya.N, [2]Ranjani. R [3]Gayathri. R , [4]Shanmuga Priya
1.Assistant professor of CSE Department .M.A.M College of Engineering and Technology, Trichy
2. Student, Department of CSE, M.A.M College of Engineering and Technology, Trichy.
3. Student, Department of CSE, M.A.M College of Engineering and Technology, Trichy.
4. Student, Department of CSE, M.A.M College of Engineering and Technology, Trichy.

Abstract— Storage systems that are networked and multi-user are becoming increasingly prevalent as the amount of data created worldwide grows at an exponential rate. Many customers, however, are still hesitant to migrate data to remote storage due to security concerns. The standard approach before data leaves the owner's premises, it must be encrypted. While this strategy is secure, it prevents the storage provider from performing storage efficiency functions such as compression and deduplication, which would allow for better resource utilization and, as a result, cheaper service costs. Data deduplication on the client side, in particular, ensures that multiple uploads of the same file use the same file amount of network traffic and storage space. Deduplication is used by a number of cloud backup businesses as well as a number of cloud services. Because encrypted data is pseudorandom and hence cannot be deduplicated, current methods are ineffective systems must choose between security and storage efficiency. We describe approaches in this research that allow for a finer-grained trade-off in data chunk similarity. And using self-destruction approach to monitor the user login activity and restore the data from cloud backup. Various deduplication strategies are investigated, with experimental results demonstrating that the suggested In real-time cloud systems, secure data chunk similarity increases results.

Index Terms- Cloud storage, Chunks of the data, Data Deduplication, Data security, Self-destruction

I. INTRODUCTION

The amount of information available today is increasing. Because cloud service providers give unlimited storage space, users seek to consume as much as they can, while suppliers are constantly looking for ways to decrease redundant data and maximize space savings. Users will obtain information based on their needs, with the majority accessing the same information several times, minimizing the cost of calculation, application hosting, content storage, and delivery. You may access Thanks to the cloud, you may access your data from anywhere at any time. Flexibility, disaster recovery, automated software updates, a pay-per-use model, and cost savings are all advantages of cloud computing. In a traditional computer setup, being in the same physical area as the data storage device is essential. The cloud eliminates this requirement. And, due to the cloud, there's no need to be in the same physical location as the data-saving technology. Each provider performs a certain role depending on the kind, giving users more or less control over their cloud. Users' cloud requirements will differ depending on how they want to use the cloud's space and resources. Cloud computing is the use of computers that connect to the Internet for processing power, storage, and applications, without the requirement for specific access points to maintain any infrastructure. Data deduplication is a method of minimizing the amount of storage space required for an organization's data. Many pieces of data in most organizations' storage systems are duplicate copies. For example, the same file could be saved in multiple locations by various users, or two or more files that aren't identical could contain a lot of the same information. Users

demand data security and confidentiality guarantees through encryption, in addition to low ownership costs and flexibility. We employ encryption for secure deduplication services to make data management scalable deduplication. Regrettably, deduplication and encryption are two technologies that are incompatible. While deduplication's goal is to discover identical data segments and save them only once, encryption's result is that two identical data segments after encryption are indistinguishable. That is to say, if customers encrypt data in a typical fashion, such as using shared authority, the cloud storage provider will be unable to do deduplication because After encryption, two identical data segments would be different. On the other hand, if consumers do not encrypt their data, secrecy cannot be guaranteed, and data is exposed to cloud storage providers that are interested. There are two types of deduplication when it comes to size: file-level deduplication, which discovers and removes redundancies between files to reduce capacity requirements, and block-level deduplication, which finds and removes redundancies between data blocks. The file can be divided into smaller portions of fixed or variable size. Using fixed-size blocks instead of variable-size blocks simplifies the computation of block borders. The AES Scheme, where the encryption key is usually the result of the hash of the data segment, and tag generation, are two techniques that have been proposed to achieve these two contradictory needs. Although encryption appears to be a viable option for achieving both It provides both secrecy and deduplication at the same time is plagued by a number of well-known flaws.Encrypting sensitive data before outsourcing to faraway servers can solve the confidentiality problem. Users demand data security and confidentiality guarantees through encryption, in addition to low ownership costs and flexibility. In this paper, we propose a shared authority for files using Deduplicated based privacy preserving authentication for cloud data storage, which achieves authentication and authorization without jeopardizing a user's personal information. The basic data component similarity is depicted in Figure 1.

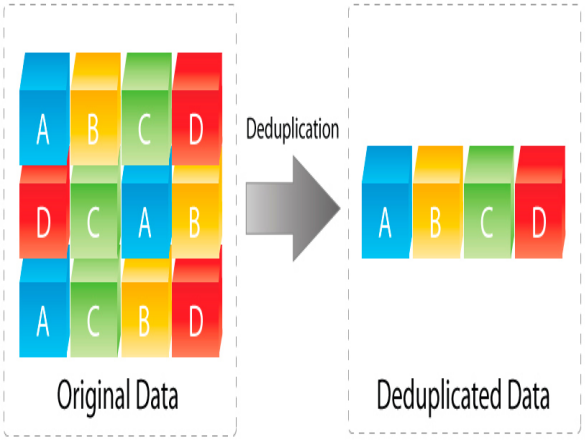

Fig 1: Data Deduplication

II. RELATED WORK

Wu, Huijun, Chen Wang, et.al,…[1] By combining an inline and a post-processing method for accurate deduplication, a Hybrid Prioritized Data Deduplication mechanism was proposed to cope with the storage system shared by applications running in co-located virtual machines or containers. HPDedup provides a fingerprint caching method in the inline deduplication phase that assesses the temporal proximity of duplicates in data streams from various VMs or apps and prioritizes cache allocation for these streams based on the estimation. To avoid disc fragmentation, HPDedup provides variable deduplication thresholds for streams based on their geographic proximity. Duplicates whose fingerprints cannot be cached due to weak temporal locality are removed from discs during the post-processing step.Although primary storage deduplication has several advantages in terms of enhancing resource sharing efficiency, the growing number of apps from different users using the same computer poses a barrier to primary storage deduplication. The cloud software stack, such as Open Stack, maps the data volumes of VMs to permanent block storage devices connected by networks in most setups.

Douglis, Fred, Abhinav Duggal, et.al… [2] The proposed system describes the logical and physical versions of our trash collecting subsystem (respectively LGC and PGC). A full description of LGC is useful for understanding its flaws as well as the fact that for many years,

this was the version utilized in the commercial product. We understand that other articles have been published after LGC was first implemented that describe alternative GC systems in depth, and we consider the technical contributions of this study to be the insights that led to the new and substantially better PGC subsystem. PGC has evolved, first with a change in container access order and then being further optimized to reduce memory usage and eliminate the need for numerous passes over the data.We compare LGC to an earlier version of PGC that has been in use at client sites for a long time, as well as a newer "phase-optimized physical GC" (PGC+) that includes further optimizations. In summary, the following are the contributions of this paper: In a deduplicating storage system, a detailed overview of two ways to garbage collection. The prior technique was replaced by a new GC algorithm whose enumeration time scales with the physical capacity of the system rather than the logical (pre-deduplication) capacity or the number of files due to changing workloads.

Frey, Davide, Anne-Marie Kermarrec, et.al,…[3] Propose a lightweight cluster-based backup system with deduplication rates comparable to single-node systems at a low computational cost and minimum memory overhead. We make two major advances as a result of this: a lightweight probabilistic node-assignment method and a new bucket-based load-balancing strategy. Product can quickly discover the servers that can deliver the highest deduplication rates for a specific data block using the former. The latter effectively distributes the load among the nodes. By combining state-of-the-art approaches with innovative additions, Product achieves deduplication rates that are close to those of a single-node system.With a two-level chunking technique, Product addresses the tradeoff between tiny and big chunk sizes. It routes and stores data using relatively big super chunks, which are made up of a number of smaller chunks that make up the deduplication information units. Second, it has two innovative additions that allow it to I achieve statefull super chunk assignments with minimal CPU and memory requirements, and (ii) balance the load on cluster nodes without deduplication performance being hampered.

Mao, Bo, Hong Jiang, et.al,…[4] Propose SAR, an SSD (solid-state drive)-Assisted Read scheme that takes advantage of SSDs' high random-read performance and deduplication-based storage systems' unique data-sharing feature by storing unique data chunks with high reference count, small size, and non-sequential characteristics in SSDs. Many read requests to HDDs are thus replaced with read requests to SSDs, considerably boosting the read performance of deduplication-based cloud storage systems. In terms of average response times, comprehensive trace-driven and VM restore assessments on the prototype implementation of SAR reveal that SAR greatly outperforms existing deduplication-based and flash-based caching techniques.The storage pool in SAR is made up of an SSD array and an HDD array. The redundancy mechanisms used by the arrays help to ensure data reliability. Only the unique data chunks are kept when the data is deduplicated, and the duplicate data chunks are replaced by pointers to the unique data chunks previously stored in the HDD array.

Jin, Keren, and Ethan L. Miller, et.al,…[5] Propose using deduplication to lower the total amount of storage required for VM disc images while still allowing VMs to share disc blocks. We conducted comprehensive assessments on multiple sets of virtual machine disc images with different chunking algorithms to test the effectiveness of deduplication. When only the locale or software configuration is changed, the amount of data stored rises extremely slowly after the first few virtual disc images, with the rate of compression degrading when different versions of an operating system or different operating systems are added. We also show that fixed-length chunks perform well, with compression rates virtually identical to variable-length chunks.We break image files into pieces to minimize their granularities in order to locate identical areas of disc pictures. Each picture file is treated as a byte stream, with boundaries identified using either a "constant" function (for fixed-size chunking) or a Rabin fingerprint (for variable-sized chunking). A chunk is merely the data between two boundaries; each image file has implicit bounds at the beginning and finish. Chunks are recognized by their SHA1 hash, which is calculated by SHA1 being applied to the chunk's contents. We don't undertake a byte-by-byte comparison to confirm that chunks with the same chunk ID are similar; instead, we presume they are.

III. DATA DEDUPLICATION TYPES

The following are the several types of data deduplication that we can examine in this chapter:

File-level de-duplication

It's also known as single-instance storage or file-level data storage. De-duplication examines a file that has to be archived or a backup that has already been stored by comparing all of its attributes against the index. The index is only updated and stored if the file is unique; otherwise, only a pointer to the existing file is referenced. As a result, only one copy of the file is saved, and relevant copies are replaced by a "stub" pointing to the original.

Block-level de-duplication

Data deduplication is done at the sub-file level, not at the block level. The file will be divided into segments, blocks, or chunks, each of which will be analyzed for previously stored data vs. redundancy, as the name implies. Assigning identification to each piece of data is the most popular approach for finding redundant data; for example, the hash algorithm generates a unique ID for each block. The core index will be compared to the unique Id in question.If the ID already exists, it signifies that only the data was already processed and stored. As a result, all that is saved in the area is a hyperlink to previously stored data. If the ID is new and does not exist, that block is unique. After storing the unique chunk, the unique ID is updated in the Index. The seller claims that the chunk size has changed. Some blocks will be of uniform size, while others will be of varying sizes.

Variable block level de-duplication

It analyses different sizes of data blocks to decrease the chances of a collision, according to storage. The differences between deduplication approaches are depicted in Figure 2.

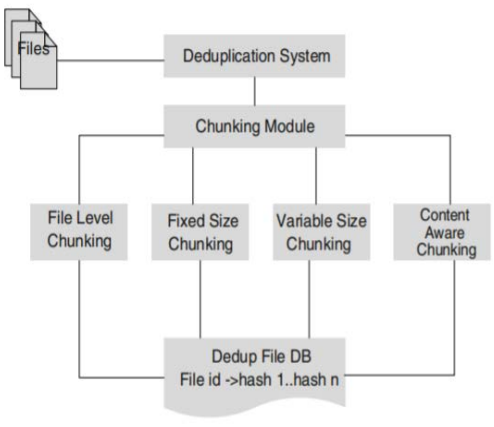

Fig 2: Deduplication schemes

IV. SECURE DEDUPLICATION ALGORITHMS

The primary purpose of this study is to examine a variety of encryption algorithms that employ deduplication strategies. The basic algorithms are as follows:

Conventional Encryption algorithm:

Although data deduplication is recognized to provide additional Because users' sensitive data is vulnerable to both outsiders and insiders, security and privacy issues occur and insider attacks. As a result, there are numerous concerns to consider when using classical encryption approaches to protect consumers' sensitive data. Data secrecy is provided by traditional encryption; however it Deduplication is incompatible with this. Different users use different keys to encrypt their data, much like with traditional encryption.As a result, different cipher text will result from identical data from various users, making data deduplication very difficult in this old manner. The algorithm's first stage is as follows:

KeyGenSE: k is a key generation algorithm that generates with the help of security parameters. I DecSE (k, M): M is the symmetric decryption algorithm that takes the secret and message M and outputs the original message M; EncSE (k, M): C is the symmetric encryption algorithm that takes the secret and message M and outputs the ciphertext C; DecSE (k, M): M is the symmetric decryption algorithm that takes the secret and ciphertext C and outputs the original message M.

Convergent Encryption Algorithm: Convergent encryption algorithms guarantee data confidentiality to customers who have outsourced data stored on public clouds. These techniques are consistent with the data deduplication process while maintaining data confidentiality. The encryption key is extracted from the message in this approach. As a result, it provides data deduplication as well, because the same file will generate the same ciphertext regardless of user, allowing for data deduplication.

The key generation algorithm KeyGenCE (M) K maps a data copy M to a convergent key K; the symmetric encryption algorithm EncCE(K,M) C takes both the convergent key K and the data copy M as inputs and outputs a

ciphertext C; the decryption algorithm DecCE(K,C) M takes both the ciphertext C and the convergent key K as inputs and outputs the original data copy M; and the (M).

Block cipher algorithm:

A block cypher is a deterministic algorithm that operates on fixed-length groups of bits, known as blocks, with an unvarying transformation given by a symmetric key in cryptography. Block cyphers are commonly utilized to encrypt bulk data and are employed as crucial elementary components in the creation of many cryptographic protocols. Multiple rounds of encryption are performed via iterated product cyphers, each of which utilizes a new sub key obtained from the original key. The DES cypher is an example of a widely used implementation of such cyphers. Substitution-permutation networks are used to classify many different block cypher implementations, such as the AES. The public's comprehension of current block cypher design was greatly aided by the disclosure of the DES cypher. It also had an impact on academic research into cryptanalytic assaults. Studies on the DES design spawned both differential and linear cryptanalysis. In addition to being secure against brute force attacks, a block cypher must be secure against a variety of attack tactics. Even the most secure block cypher is only good for encrypting a single block with a set key. A variety of modes of operation have been developed to enable for their repeated usage in a secure manner, with the most common security goals of confidentiality and authenticity in mind.Other cryptographic protocols, such as universal hash functions and pseudo-random number generators, may use block cyphers as building blocks.

A substitution and permutation network (SPN) is a sort of iterated block cypher that takes a block of plaintext and the key as inputs and executes numerous alternating rounds—a substitution stage followed by a permutation stage—to produce each block of cypher text output. Shannon's perplexity is caused by the non-linear replacement stage, which mixes the key bits with those of the plaintext. Following that, the linear permutation stage dissipates redundancies, resulting in diffusion. A substitution box (S-box) replaces one set of input bits with another set of output bits. To ensure invertibility, this substitution must be one-to-one (hence decryption). A secure S-box will have the property that changing one input bit affects roughly half of the output bits on average, displaying the avalanche effect—that is, each output bit will be dependent on every input bit.

Variable chunk similarity:

Because the number of IDs to be processed increases dramatically, it necessitates greater processing capacity than file deduplication. As a result, its index for tracking individual iterations grows significantly. The use of variable length blocks consumes much more resources. Furthermore, hash collisions occur when the same hash number is created for two separate data pieces. Because the hash number already exists in the index, the system will not preserve the new data if this happens.

The procedure is as follows:

BlockTag (FileBlock) - As a file block Tag, it computes the hash of the File block.

DupCheckReq (Token) - It asks the Storage Server to check the file block for duplicates.

FileUploadReq (FileBlockID, FileBlock, Token) – If the file block is unique, it uploads the file data to the Storage Server and changes the file block Token.

TokenGen(File Block, User ID) – the process loads the associated privilege keys of the user and generates token; FileBlock Encrypt(Fileblock) – it encrypts the file block with Convergent Encryption, where the convergent key is from the SHA Hashing of the file block; FileBlock Encrypt(Fileblock) – it encrypts the file block with Convergent Encryption, where the convergent key is

It stores the FileBlock on disc and changes the Mapping with FileBlockStore(FileBlockID, FileBlock, Token). Figure 3 depicts the variable chunk similarity level deduplication.

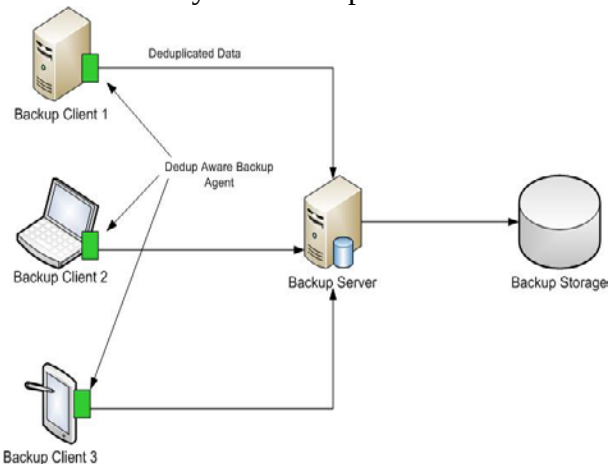

Figure 3: Backup server with variable chunk similarity

The following is the mathematical model:
Assume that S is the system object.
It consists of the following components: S={U,F,CSP}
U stands for the number of users. U={u1,u2,u3,…..un}
F = number of files F={f1,f2,f3,…..fn}
B stands for the number of blocks. B{B1,B2,…,Bn}
CSP={C,PF,V,POW}
C=challenge
PF = CSP proof
TPA TPATPATPATPATPATPATPA T
POW stands for proof of ownership.
CSP stands for Cloud Service Provider.
CSP={PF,F} PF=proof F=files
Figure 4 depicts the suggested architecture.

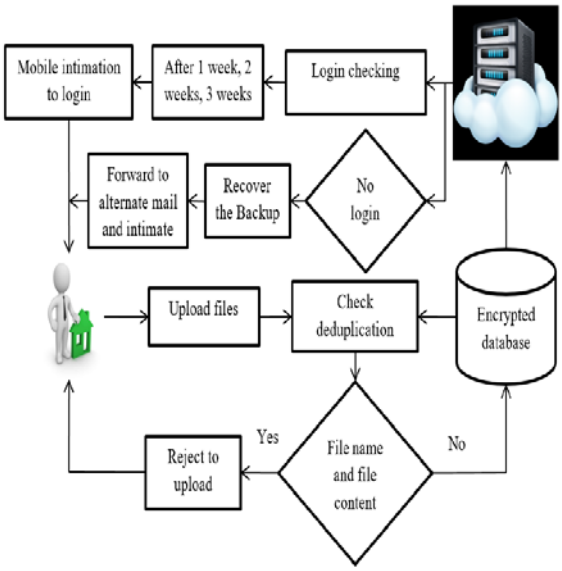

Fig 4: Proposed framework

Cloud storage framework:

Cloud computing and storage solutions give consumers and businesses the ability to store and process data in privately owned or third-party data centers that are often located far away from the user–ranging from across a city to across the globe. To achieve coherence, cloud computing relies on resource sharing. We can have two categories of consumers in this framework: data owners and data providers. A cloud service owner is a person or organization who legally owns a cloud service.The cloud service owner might be either the cloud consumer or the cloud provider who owns the cloud where the cloud service is located. Users are given storage space by the cloud service provider. Multiple data owners can share storage space. Owners of data can save files in a storage system for later use.

File encryption:

Encryption is the most efficient method of ensuring data security. You must have access to a secret key or password to decode an encrypted file in order to read it. Plain text refers to unencrypted data, while cypher text refers to encrypted data. Asymmetric encryption (also known as public-key encryption) and symmetric encryption are the two basic forms of encryption.Using a single key technique, we can use symmetric encryption to encrypt data files. Symmetric key algorithms are cryptographic algorithms that use the same cryptographic keys for plaintext encryption and cypher text decoding. The keys could be identical, or there could be a simple transformation that connects them. In practice, the keys constitute a shared secret between two or more parties that can be utilized to maintain a secure data transmission. Data that has been encrypted can be stored on a cloud server.

Deduplication checking:

Data deduplication is a specific data compression technique for removing multiple copies of repeated data in computing. Intelligent (data) compression and single-instance (data) storage are two words that are related and partly synonymous. This technique is used to increase storage usage and can also be used to reduce the number of bytes that must be sent over a network. During the deduplication process, unique pieces of data, also known as byte patterns, are discovered and stored.Other chunks are compared to the stored copy as the analysis progresses, and anytime a match is found, the superfluous chunk is replaced with a short reference to the saved chunk. We can check the files in this module by comparing the file name to the file contents. Encrypted files are broken up into pieces. When uploading files, the service provider examines the pieces. To save storage space in the cloud system, the data owner just uploads the original file. Text, document, and image files can all be deduplicated.

Alert system:

In this system, we can create an application for a weekly alert system. If there is no access after four weeks, the files are automatically distributed to alternate mail and mobile

numbers that were provided at the time of registration. The server can store a large amount of data and make it available to other users.

Backup recovery approach:

Each user's access time can be checked by the administrator. If a user logs in to the system, activity is recorded in the database. In addition, each user's access is monitored. If a user's access is paused for more than three days, the administrator will send an alert to the user's registered cellphone number. Finally, if access to the storage system is denied, a backup is created. Also, flush the storage space and save it for future usage on the server.

V. EXPERIMENTAL RESULTS

The proposed approach is evaluated in terms of preserving storage and is implemented in real-time situations. Figure 4 depicts the proposed outcome.

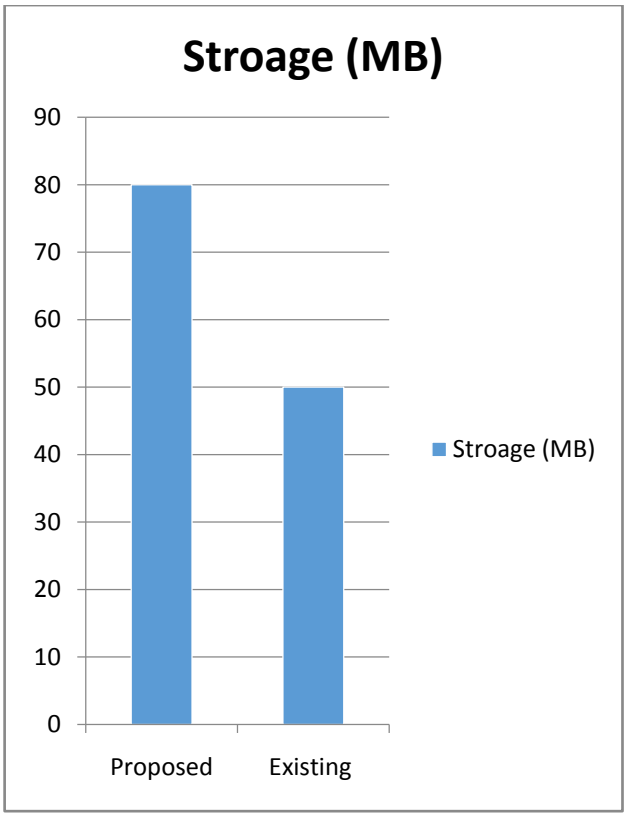

Fig 5: Experimental Results

The proposed solution preserves up to 80% of storage while maintaining security.

VI. CONCLUSION

This paper focused at distributed deduplication systems and shared authority outsourced data using an encryption technique to improve data dependability while respecting user privacy. The constructions were built to withstand the elements data deduplication at the file and block levels. The uniformity and integrity of the tags were ensured. We used the block cypher scheme with variable chunk similarity to create our deduplication systems, and we demonstrated that it had a low when compared to the network transmission overhead in typical upload/download operations; encoding/decoding overhead is higher. Because of this, wrapped values are swapped during transmission, data anonymity is obtained. Access requests enhance user privacy by informing the cloud server about the user's access preferences in a secret manner.

REFERENCES

[1] Wu, Huijun, Chen Wang, Yinjin Fu, SherifSakr, Liming Zhu, and Kai Lu. "Hpdedup: A hybrid prioritized data deduplication mechanism for primary storage in the cloud." arXiv preprint arXiv:1702.08153 (2017).

[2] Douglis, Fred, AbhinavDuggal, Philip Shilane, Tony Wong, Shiqin Yan, and FabianoBotelho. "The logic of physical garbage collection in deduplicating storage." In 15th {USENIX} Conference on File and Storage Technologies ({FAST} 17), pp. 29-44. 2017.

[3] Frey, Davide, Anne-Marie Kermarrec, and KonstantinosKloudas. "Probabilistic deduplication for cluster-based storage systems." In Proceedings of the Third ACM Symposium on Cloud Computing, p. 17. ACM, 2012.

[4] Mao, Bo, Hong Jiang, Suzhen Wu, Yinjin Fu, and Lei Tian. "Read-performance optimization for deduplication-based storage systems in the cloud." ACM Transactions on Storage (TOS) 10, no. 2 (2014): 6.

[5] Jin, Keren, and Ethan L. Miller. "The effectiveness of deduplication on virtual machine disk images." In Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, p. 7. ACM, 2009.

[6] Paulo, João, and José Pereira. "Efficient deduplication in a distributed primary storage infrastructure." ACM Transactions on Storage (TOS) 12, no. 4 (2016): 20.

[7] B. Li, E. Mazur, Y. Diao, A. McGregor and P. Shenoy, “A platform for scalable one-pass analytics using mapreduce,” in: Proceedings of

the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), 2011, pp. 985-996.

[8] R. Kienzler, R. Bruggmann, A. Ranganathan and N. Tatbul, "Stream as you go: The case for incremental data access and processing in the cloud," IEEE ICDE International Workshop on Data Management in the Cloud (DMC'12), 2012

[9] C. Olston, G. Chiou, "Nova: Continuous pig/hadoop workflows," Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), pp. 1081-1090, 2011.

[10] K.H. Lee, Y.J. Lee, H. Choi, Y.D. Chung and B. Moon, "Parallel data processing with mapreduce: A survey," ACM SIGMOD Record 40(4): 11-20, 2012.

[11] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev,"Message-locked encryption for lock-dependent messages," in Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, 2013, pp. 374–391.

[12] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, "Secure and efficient cloud data Deduplication with randomized tag," IEEE Trans. Information Forensics and Security, vol. PP, no. 99,

[13] Y. Zhou, D. Feng, W. Xia, M. Fu, F. Huang, Y. Zhang, and C. Li, "Secdep: A user-aware efficient fine-grained secure Deduplication scheme with multi-level key management," in IEEE 31st Symposium on Mass Storage Systems and Technologies, MSST 2015, Santa Clara, CA, USA, May 30 - June 5, 2015, 2015, pp. 1–14.

[14] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," IEEE Trans. Big Data, vol. 2, no. 2, pp. 138–150, 2016.

[15] R. Bhaskar, S. Guha, S. Laxman, and P. Naldurg, "Verito: A practical system for transparency and accountability in virtual economies," in 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013, 2013.

[16] D. T. Meyer, and W. J. Bolosky, "A study of practical deduplication," Proc. USENIX Conference on File and Storage Technologies 2011

[17] W. K. Ng, W. Wen, and H. Zhu, "Private data deduplication protocols in cloud storage," Proc. ACM SAC'12, 2012.

[18] N. Baracaldo, E. Androulaki, J. Glider, A. Sorniotti, "Reconciling end-to-end confidentiality and data reduction in cloud storage," Proc. ACM Workshop on Cloud Computing Security, pp. 21–32, 2014.

[19] P. Anderson, L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," Proc. USENIX LISA, 2010.

[20] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," IEEE Transactions on Parallel and Distributed Sytems, Vol. 25, No. 6, 2014.

[21] J. Li, Y. K. Li, X. Chen, P. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 5, pp. 1206–1216, 2015.

[22] M. Bellare, S. Keelveedhi, T. Ristenpart, "DupLESS: Server aided encryption for deduplicated storage," Proc. USENIX Security Symposium, 2013.

[23] M. Bellare, S. Keelveedhi, "Interactive message-locked encryption and secure deduplication," Proc. PKC 2015, pp. 516–538, 2015.

[24] L. Mingqiang, C. Qin, P.P.C. Lee, and J. Li, "Convergent Dispersal: Toward Storage-Efficient Security in a Cloud-of- Clouds," Proc. USENIX Conference on Hot Topics in Storage and File Systems, 2014.

[25] J. Li, X. Chen, X. Huang, S. Tang, Y. Xiang, M. Hassan, and A. Alelaiwi, "Secure Distributed Deduplication Systems with Improved Reliability," IEEE Transactions on Computer, Vol. 64, No. 2, pp. 3569–3579, 2015.