Distributed Data into Deduplication System with Security to Improve Reliability

Mrs. A. S. Hambarde
Department of Computer Engineering
K. J. College of Engineering Pune, India

Leena Rade
Department of Computer Engineering
K. J. College of Engineering Pune, India

Kajal Kamthe
Department of Computer Engineering
K. J. College of Engineering Pune, India

Priti Andhale
Department of Computer Engineering
K. J. College of Engineering Pune, India

Kajal Memane
Department of Computer Engineering
K. J. College of Engineering Pune, India

Abstract

Data deduplication is a technique that has been widely used in cloud storage reduced storage space and uploads bandwidth. Deduplication system improves storage utilization while reducing reliability. This paper makes the first attempt to formalize the notion of distributed reliable deduplication system. Proposed a new deduplication system with distributed data. This system provides security for confidential data and also improves reliability. This can be achieved by secret sharing, proof of ownership and decoy techniques.

Keywords: POW, SSS (Secret Sharing Schemes), AES (Advance encryption standard)

I. INTRODUCTION

As volume of data goes on increasing day by day on network, for that cloud storage system is used. But still cloud system has many challenges to face regarding storage of data. Cloud storage system provides highly available storage and parallel computing at low cost with the help of authorized access to every user.

Main challenge face by cloud is storage service management of duplication. This duplication of data having wastage of storage space to overcome this problem deduplication technique is used, which will check duplicate copies of data; if it is found then it will eliminate these duplicate copies of data to reduce storage space and upload bandwidth. There is only one copy of data will be stored on cloud and that copy will be access by many users.

Second main challenge to cloud is security data of user. Security requirement of data confidentiality and tag consistency. This can be achieved by introducing secret sharing in distributed storage system instead of convergent encryption.

For authorized user to provide their ownership of data copies to storage system server we used POW that is proof of ownership. This is an interactive algorithm which is run by power and verifier. It is used in content distribution network, where an attacker does not know entire files but has accomplices who have file. Accomplices help attacker to obtain file, subject to constraint that they must sent fewer bits than initial min-entropy of file to attacker.

Also for privacy and security purpose we introduced decoy technique. Decoy is the bogus information such as honeypots, honeyfiles or documents that can be generated on demand and serve as information of while detecting on unauthorized access. And also provide poison to ex-filtrated information of thief. This decoy information automatically returns by cloud and deliver in the form of normal information. But owner of file can be identifying by reading that this is bogus information. In this way true data will be remain secure.

II. LITERATURE SURVEY

This is literature survey, which helps to improve system. So this literature survey says that "Reclaiming Space from Duplicate Files in a Serverless Distributed File System" John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, Marvin Theimer Microsoft Research in 2002, this paper helps deduplication for reclaiming space. This can be done with convergent encryption, also it uses SALAD technique for Aggregating file content and location information in decentralized, reliable, fault tolerant manner. Encryption prevents file hosts from unauthorized viewing of file content. Drawbacks with convergent encryption which deliberately leaks a controlled amount of information. Also these techniques are not comfortable with distributed storage system instead of farsite system.

"Secure Data Deduplication" Mark W. Storer Kevin Greenan Darrell D.E. Long Ethan L. Miller Storage System Research Center University of California, Santa Cruz" in 2008, helps to provide data security and space efficiency with 2 models. -

1) Authorized model, each user has asymmetric key that is private to that user. A certificate authority exists to facilitate trusted distribution of public keys; users are able to generate cryptographically sound encryption keys.
2) Anonymous model, goal is to hide identities of both authors and readers.

Main drawback is that anonymous data store is that both will be behaved and malicious users are anonymous.

"Fast and Secure Laptop Backups with Encrypted De-duplication" Paul Anderson university of Edinburgh Le Zhang University of Edinburgh in 2010, provided an algorithm of deduplication and convergent encryption which works for any type of system but it is particularly appropriate for laptops where connection speed and network availability are bigger issues than processing time. Algorithm uses hash function used to return unique key for data block based on contents of data. In some cases, it is necessary to store additional metadata or reference count to keep track of multiple owners but no need to store multiple copies. Each block new has a separate encrypted key some mechanism is needed to each owner to record and retrieve keys associated with their dat blocks. But some disclosure of information, if a user has copy of file, it is possible to tell whether or not some other user also has copy of same file.

"DupLESS: Server Aided Encryption for Deduplicated Storage" Mihir Bellare University of California, San Diego Sriram Keelveedh University of California, San Diego Thomas Ristenpart university of Wisconsin Madison in 2013, provide secure deduplicated storage resisting brute force attacks and realize it in system called DupLESS. Implement DupLESS as simple to use command line that support Dropbox and Google drive as storage service. They implement KS (Key Server) protocol with it's 2 versions, first being able to run on top of existing web server. Second to optimized for latency and capable of servicing request at close to round trip time of network. Main goal is to protect confidentiality of client data. It means that system will be backwards compatible work within exists SS APIs make no assumptions about implementation details of SS.

"Convergent Dispersal: Toward Storage – Efficient Security in a Cloud of clouds" Mingqiang Li, Chuan Qin, Patrick P. C. Lee, Jin Li the Chinese University of Hong Kong, Guangzhou university in 2014, In cloud of cloud we disperse data with a certain degree of redundancy across multiple independent clouds managed by different vendors, such that stored data can always be available even if a subset of clouds becomes in accessible. Main idea is to replace random information derived with deterministic cryptographic hash information derived from original dat. They analyze deduplication efficiencies and evaluate their performance. But the problem regarding disperse convergent difficult to implement in real life dispersed setting.

III. PROPOSED SYSTEM

Also propose Dekey, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers. Dekey using Ramp secret sharing scheme and demonstrate that Dekey incurs limited overhead in realistic environments, as propose a new construction called Dekey which provides efficiency and reliability guarantees for convergent key management on both user and cloud storage sides. For this cloud security and reliability, we are implementing four algorithms-

A. SHA 256

This algorithm is used to hash key generation and to check file deduplication. Also having fixed size hash key value. Algorithm follows steps as-

1) Derive set of round keys from ciphertext
2) Appending padding bits. Original message is padded(extended) to that it's length (in bits) is congruent to 448, modulo 512
3) Appending length 64 bits are appended to the end of padded message to indicate length of original message in bytes
4) Preparing processing function
5) Preparing processing constants
6) Initializing buffers
7) Processing message in 512 bits blocks

B. AES (Advance encryption standard)

This algorithm is used for encryption and decryption purposed. AES is a symmetric block cipher, means it uses same key for both encryption and decryption. AES accept block size of 128 bits and a choice of 3 layers 128,192,256. Half of the data is used to modify other half and then halves are swapped. In this case, data block is processed in parallel during each round using substitution and permutation. This nature of AES allows for a fast software implementation of algorithm. AES is not scalable but it is faster for encryption and decryption operation. Also power consumption is low with excellent security. Simulation speed and hardware and software implementation is also faster. Algorithm follows steps as-

1) Derive the set of round keys from cipher key
2) Initialize state array with block data (plaintext)
3) Add initial round key to starting state array
4) Perform 9 rounds of state manipulation
5) Perform 10^{th} round and final round
6) Copy final state array out as encrypted data (ciphertext)

C. *HMAC- SHA*

This algorithm is used to token generation which is used for proof of ownership. Algorithm follows steps as-

1) Append zeros to left end of K to create b-bit. String K+(for example K is having length 160 bits and b=512, then K will be appended with 44 zero bytes 0x00)
2) XOR(bitwise exclusive OR) K+ with ipad to produce b bit block Si.
3) Append M to Si
4) Apply H to stream guaranteed in step 3
5) XOR K+ with opad to produce b bit block S0.
6) Append hash result from step 4 to S0
7) Apply H to stream guaranteed instep 6 and output result

D. *SSS (Secret Sharing Schemes)*

This algorithm is used for splitting and merging of uploaded file.
1) Sharing algorithm-
Share(m)-> (S1, S2, S3,…, Sn pub)
Secrets S1,…, Sn are distributed securely among servers 1 through n and pub is public share (we include pub for sake of generally but observe that it is often empty. If pub is present we assume that it is authenticated, so no one change it. It's just published in sky)
2) Recovery algorithm-
Rec(S'1, S'2,….., S'n, pub)=m'
Correctness properly of algorithm says that for any message m, Rec (share (m))=M.
Now using this algorithm we are constructing our architecture.

IV. ARCHITECTURE

As system having 4 algorithms, model is also divided into 4 modules, such as –

A. *Hash Key Generation*

SHA 256 algorithm is used in this module. This algorithm creates fixed size of hash key value. In some cases, there is need to appending some padding bits in original length of message or string to extend string in bytes. Also prepared processing function and constants with initializing buffers. This key value for each entry will be stored in database.

B. *Encryption of File*

AES is used here, it is symmetric hence it uses same key for encryption and decryption. As AES accept 128 bits block size , so that it performs 9 rounds on plaintext data at, at the time of 10^{th} round we will get cipher text data. It means encrypted format of file.

Now user will be able to upload the file on public cloud with the help of file token.

C. *Token Generation*

HMAC- SHA is used for token generation. This token of file is nothing but the
File ID + Hash key value. At the time of uploading file, this token is generated. But this token is useful at the time of downloading of file to know who the owner of this file is, this technique is called as proof of ownership. Also for secret sharing of file, this token is needed.

D. *Distribution or Splitting of File*

Secret sharing schemes is used for splitting and merging. In this phase, file distributed on different server through splitting. At the time of using AES algorithm file is already divided into 3 files -. des, .enc, iv.ene. And further each file from these 3 files divided into number of server. In this way splitting of file is done. And same reverse concept is applied for merging. Due to AES complexity increases for partition of file, but this will also provide faster execution time and high security for confidential data.

This is the architecture diagram of proposed system, which will provide security and reliability in distributed environment on cloud.

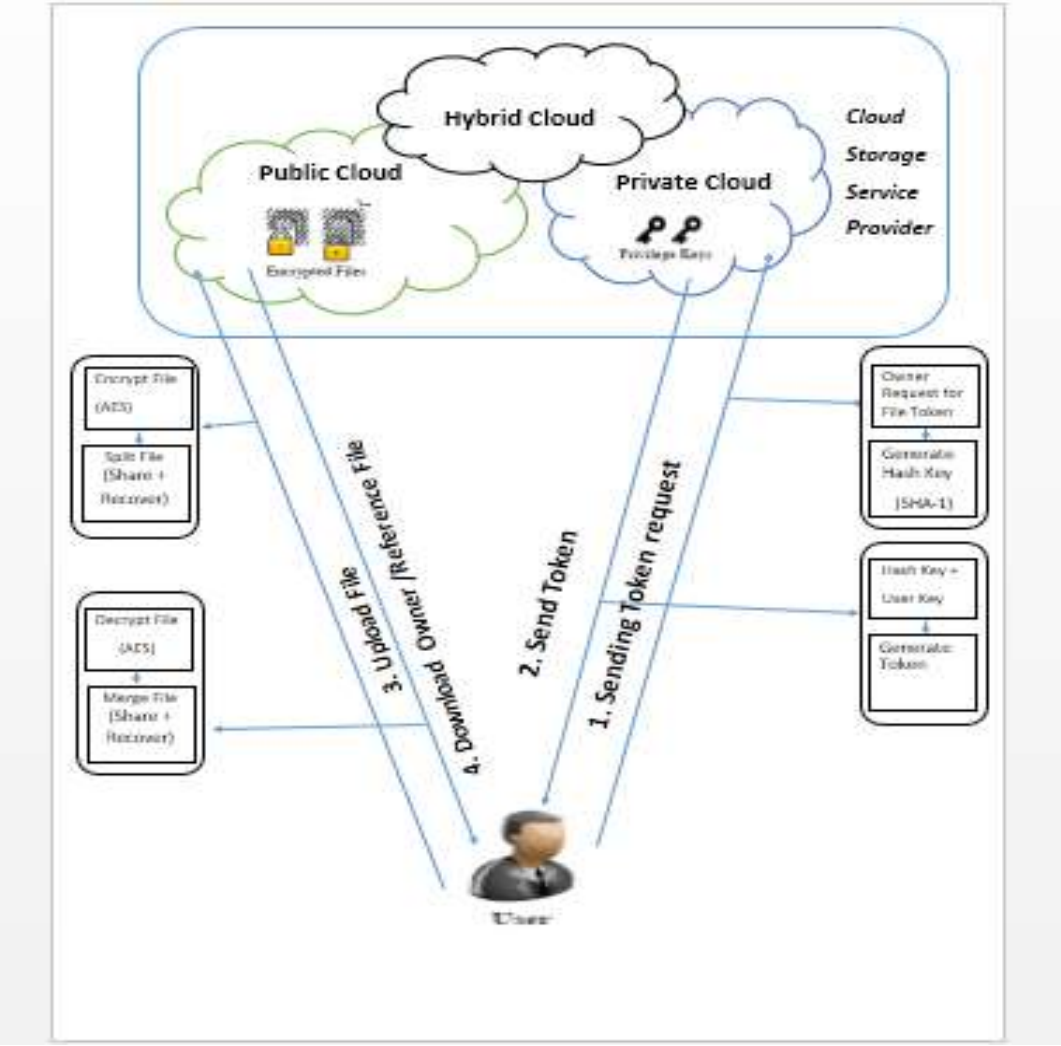

Fig. 1:

V. RESULT

System is comfortable in distributed environment. Implementing these techniques to check deduplication of file in distributed environment, also providing high security with improved reliability for confidential data. Right now we are working with text file and applying these techniques to text file. Proposed system is successfully giving results for this text file.

Fig. 2: Memory GC chart

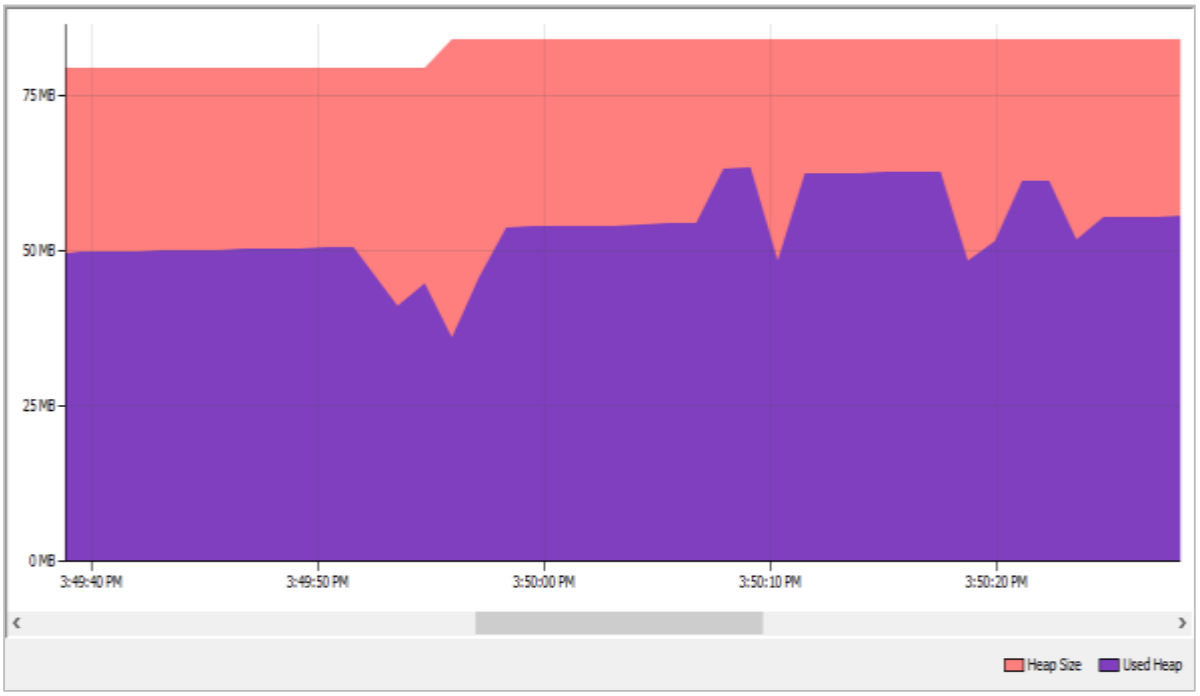

Fig. 3: Memory heap chart

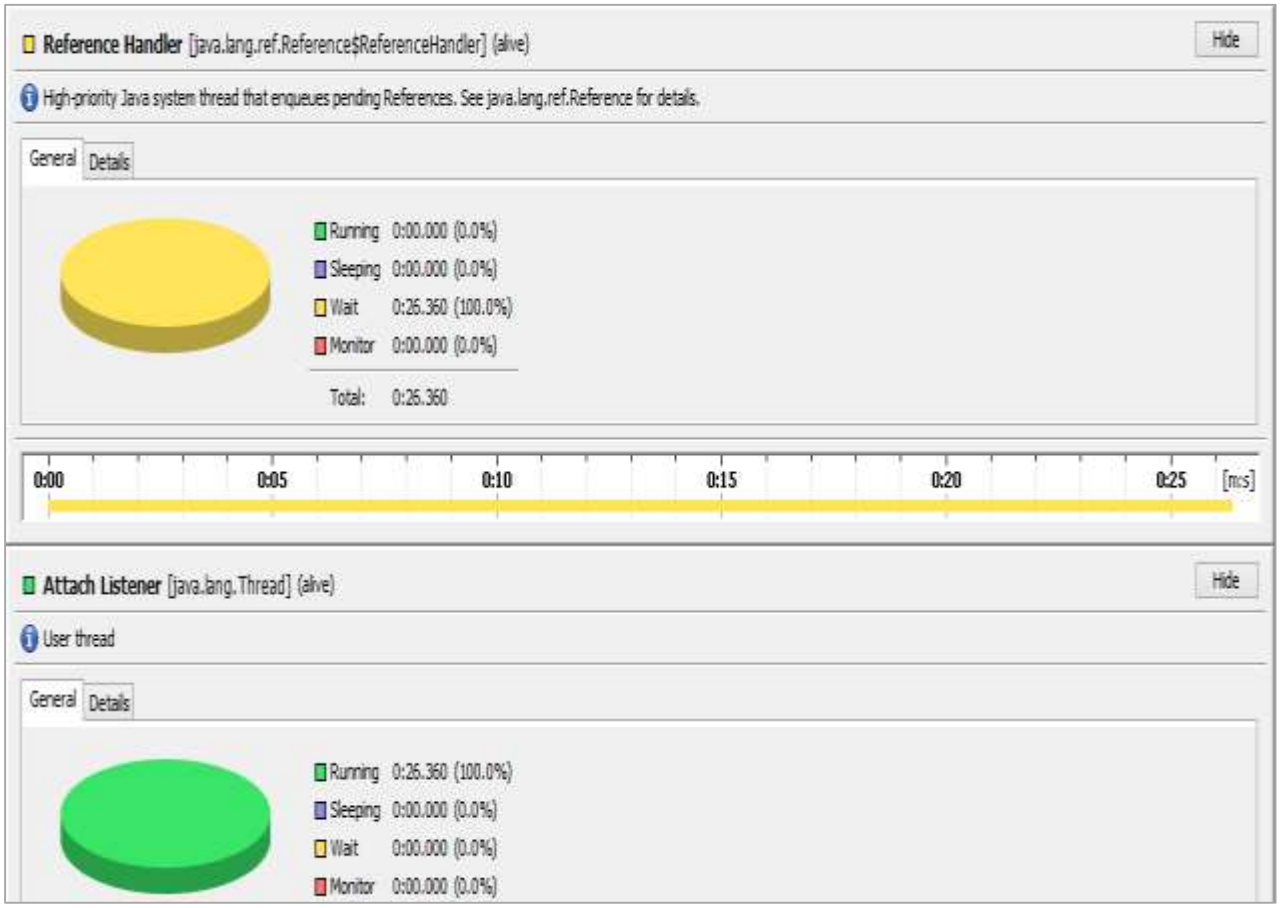

Fig. 4: Thread details

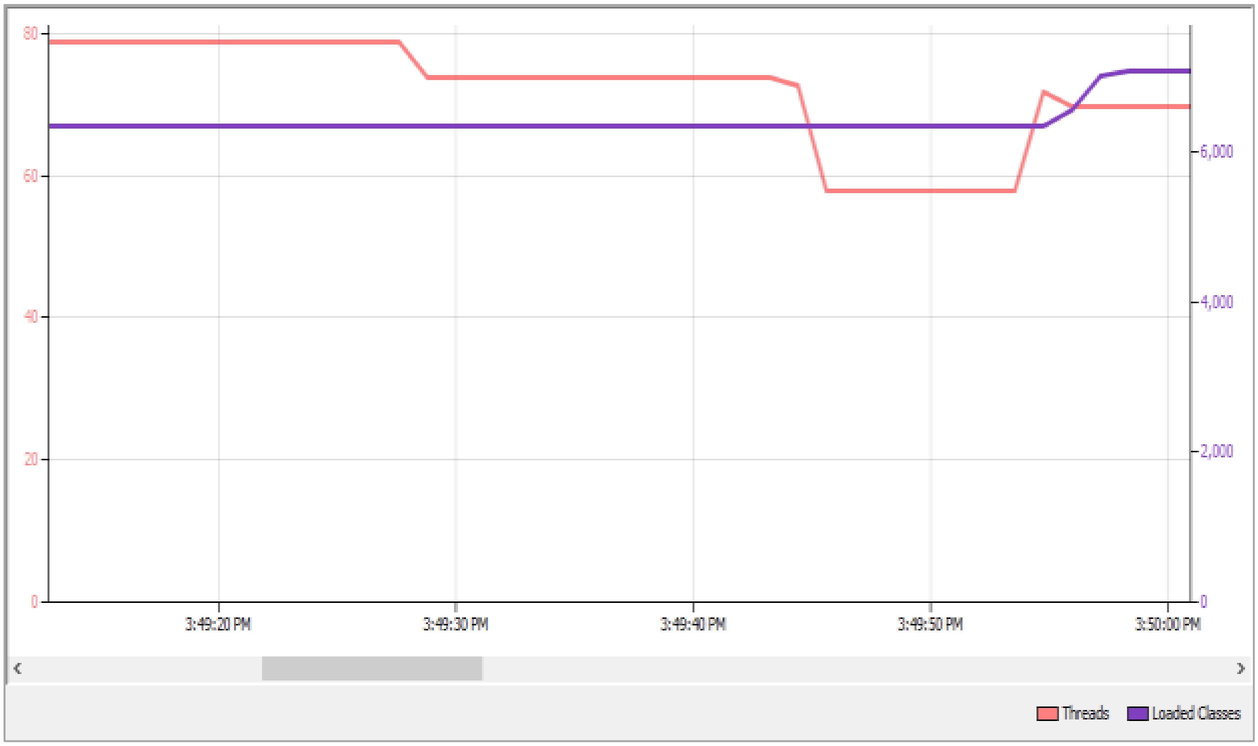

Fig. 5: Thread statistics

Table – 1
Threads

Thread	Running ▾	Sleeping	Wait	Monitor	Total
Attach Listener	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
Signal Dispatcher	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
Grizzly-kernel-thread(1)	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
Grizzly-kernel-thread(1)	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
Grizzly-kernel-thread(1)	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
Grizzly-kernel-thread(1)	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
Grizzly-kernel-thread(1)	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
RMI TCP Accept-8686	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
DestroyJavaVM	32.486 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	32.486
pool-1-thread-1	0.983 (3.0%)	0.0 (0.0%)	31.503 (96.9%)	0.0 (0.0%)	32.486
DynamicReloader	0.328 (1.0%)	0.0 (0.0%)	32.158 (98.9%)	0.0 (0.0%)	32.486
Grizzly	0.109 (0.3%)	0.0 (0.0%)	32.377 (99.6%)	0.0 (0.0%)	32.486
AutoDeployer	0.109 (0.3%)	0.0 (0.0%)	32.377 (99.6%)	0.0 (0.0%)	32.486
Reference Handler	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
Finalizer	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
FelixDispatchQueue	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
FelixStartLevel	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
FelixPackageAdmin	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
Thread-6	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
Thread-12	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
admin-thread-pool-4848(4)	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
Thread-8	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486
admin-thread-pool-4848(2)	0.0 (0.0%)	0.0 (0.0%)	32.486 (100.0%)	0.0 (0.0%)	32.486

Table – 2
Threads timeline chart

Fig. 6: Encryption performance

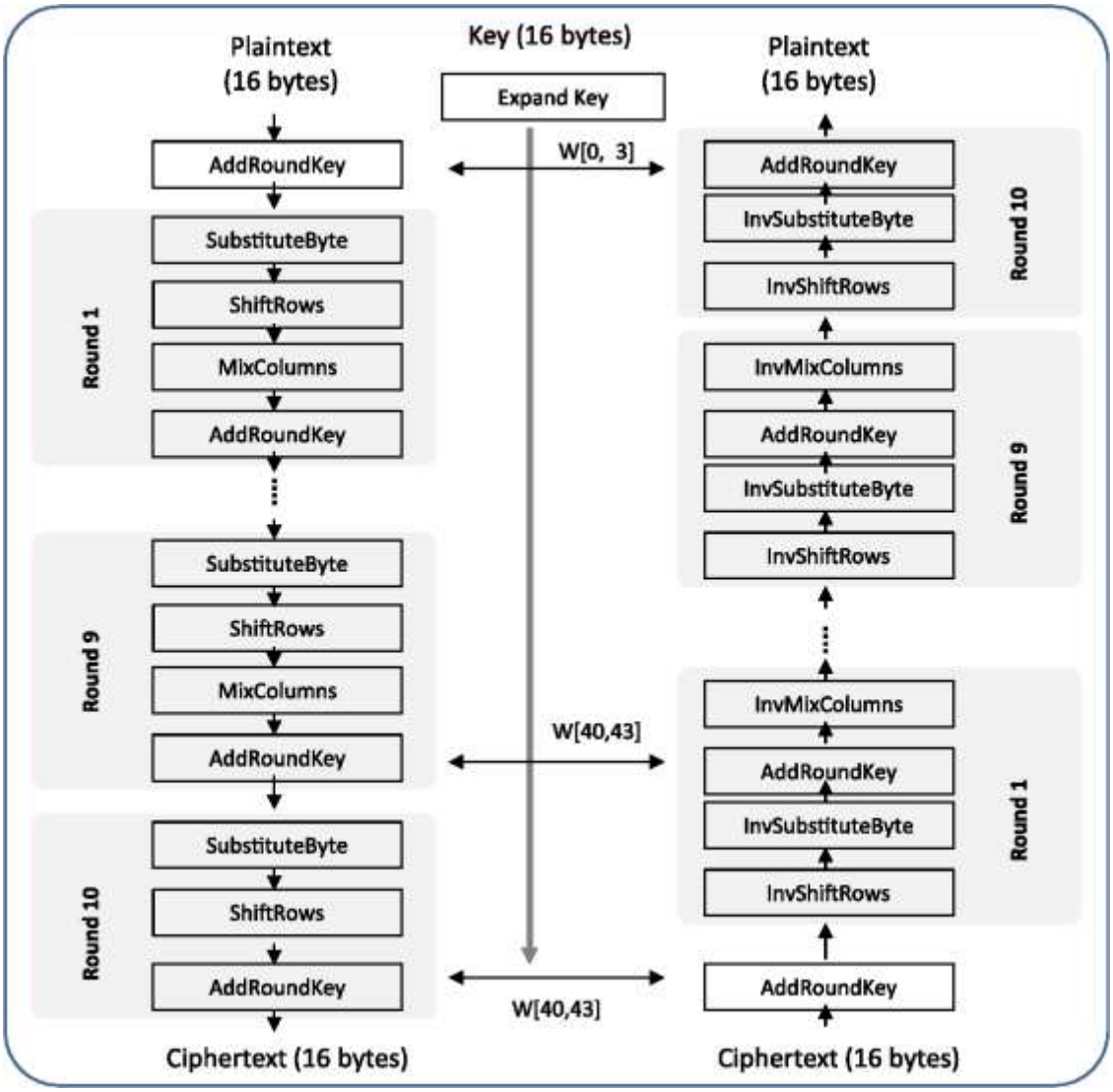

Fig. 7:

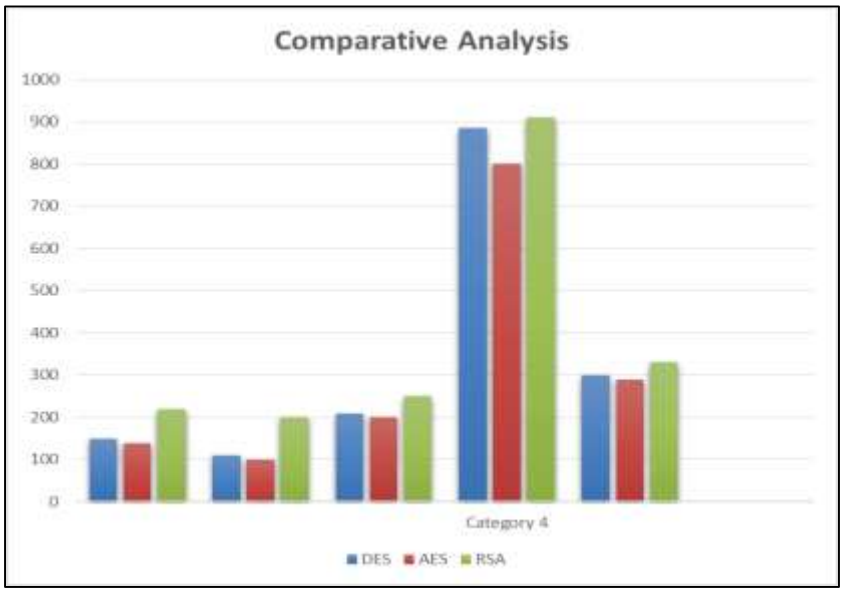

Fig. 8: comparative analysis

By analyzing this figure, it shows buffer size usage by AES, DES, and RSA algorithm buffer size usage are highest for all sizes of document file.

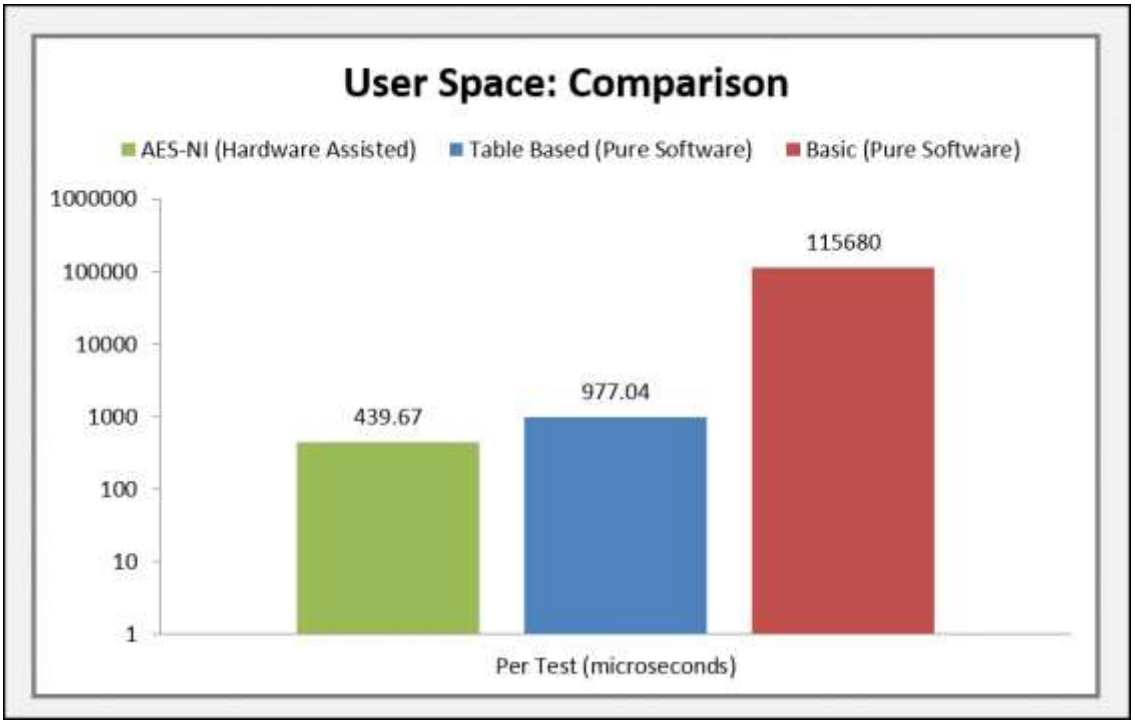

Fig. 9: user space: comparison

VI. CONCLUSION

Proposed distributed deduplication systems to improve the reliability of data while achieving the confidentiality of the users' outsourced data by using four algorithms with high security. Four modules were proposed to support file-level and fine-grained data deduplication. The security of tag consistency and integrity were achieved with the help of proof of ownership. Implemented deduplication system and demonstrated that it incurs small encoding/decoding overhead compared to the network transmission overhead in regular upload/download operations.

REFERENCES

[1] "Secure Distributed Deduplication Systems with Improved Reliability" Jin Li, Xiaofeng Chen, Xinyi Huang, Shaohua Tang and Yang Xiang Senior Member, IEEE and Mohammad Mehedi Hassan Member, IEEE and Abdulhameed Alelaiwi Member, IEEE,2015
[2] M. Li, C. Qin, P. P. C. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds," in The 6th USENIX Workshop on Hot Topics in Storage and File Systems, 2014.
[3] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Serveraided encryption for deduplicated storage," in USENIX SecuritySymposium, 2013.
[4] "DupLESS:Server-Aided Encryption for DeduplicatedStorage∗"inunsenix security symposium
[5] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," in Proc. of USENIX LISA, 2010.
[6] M. W. Storer, K. Greenan, D. D. E. Long, and E. L. Miller, "Secure data deduplication," in Proc. of StorageSS, 2008.
[7] 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, 2002, pp. 617–624.
[8] "Convergent Dispersal: Toward Storage-Efficient Security in a Cloud-of-Clouds," in EUROCRYPT, 2013, pp. 296–312.
[9] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: the least-authority filesystem," in Proc. of ACM StorageSS, 2008.
[10] 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 3rd International Workshop on Security in Cloud Computing, 2011.
[11] C. Liu, Y. Gu, L. Sun, B. Yan, and D. Wang, "R-admad: High reliability provision for large-scale de-duplication archival storage systems," in Proceedings of the 23rd international conference on Supercomputing, pp. 370–379.
[12] Amazon, "Case Studies," https://aws.amazon.com/solutions/casestudies/# backup.