International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

AN IMPROVED ALGORITHM FOR SECURED CLOUD STORAGE

Dr V SASIKALA[1] , S LOGESHWARA GUPTA[2] , S HARINI[3] , S HEMA[4]

1 ASSISTANT PROFESSOR

2,3,4 FINAL YEAR STUDENTS

DEPARTMENT AND COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING THIRUKUVALAI

ABSTRACT

Cloud storage service has shown its great power and wide popularity fundamental support for rapid development of cloud computing,due to management negligence and harmful attack, there still lie massive security event that lead to aggregate of sensitive data leakage at cloud storage layer. From the perspective of protecting cloud data confidentiality, this paper proposed a cloud Secure Storage Mechanism named CSSM. To avoid data breach at the storage layer, CSSM integrated data dispersion and distributed storage to realize encrypted, chucked and distributed storage. The experimental results specify that proposed appliance is not only suitable for ensuring the data security at storage layer from leakage, but also can store huge amount of cloud data effectively without imposing too much time overhead.

INDEX TERMS

Cloud computing, Data encryption, Data dispersion, Hashing, OTP validation, Key management, Storage security.

INTRODUCTION

Cloud computing has shown extraordinary development in recent decades. When the storage as a service, it occupies the center stage and backbone for numerous operations, like pattern recognition, image forensic and phony discovery. As a outgrowth, larger volumes of data will be a part of the cloud area. In the cloud industry, Amazon Web Service(AWS) has come the de facto standard. As the core part of the OpenStack, Swift has come one of the most trendy cloud storehouse medium.

Although, Openstack Swift medium still faces numerous real security pitfalls while on condition that accessible services. According to Cloud Security Alliance's top trouble case examination report issue in 2018, two thirds of the cases will origin stoner data leakage, substantially due to operation negligence and vicious attacks, under dereliction configuration, OpenStack Swift medium generally stores data in plain textbook for the end of performance. It may lead unauthorized access to stoner data at storehouse subcaste. Security Report issued by Openstack Vulnerability Management Team VMT, the Swift medium may blunder stoner data or configuration information in virtue of security vulnerabilities.

Shah et al suggest a cloud- acquainted data security storehouse medium under the frame of ApacheSpark, which cover data leakage and improves the security of Apache Spark frame. To save stoner data on the cloud, different encryption schemes have been espoused to avoid information

leakage during machine literacy process. still, researches bear secure crucial operation mechanisms to cover cryptographic accoutrements exposure. Zerfos etal. make a secure distributed storehouse system grounded on Hadoop system, which put by the confidentiality of cloud data through data dissipation and encryption.It acts the data decryption and assembly tasks before reading data. To cover the keys from spirit stolen, this fashion requires crucial cache garcon and all keys should be stored in memory only. Several approaches establish independent third party to handle the key. It's suppose that third parties stay trusted. Although, the supposition cloud not always exists in the real cloud storehouse surroundings.

Wang et al introduce a data sequestration maintain a suggestion for detector- cloud system, edge computing and discriminational storehouse system. For this, stoner data would be divided into several corridor and collect in original, edge and cloud subcaste independently. But the scheme figure out on the point of data from wireless detector networks, and essential professed druggies to handle the edge waiters. To enhance effectiveness and reduce the redundancy, Zheng etal. produce a cloud data deduplication scheme to descry and cancel identical stoner data in the cloud. Although, from the perspective of precluding data loss because of disaster, a certain number of clones should be transferred to multiple regions.

In a word, to save cloud data from leakage at storehouse subcaste, this paper presents CSSM, a Cloud Secure Storage Medium. CSSM combines data collect with data encryption, In such a way that large- scale cloud data and keys would be save in chunked cipher textbooks. On these terms, stoner word and secret sharing are establish to further cover keys security. We negotiate CSSM grounded on OpenStack Swift medium and made different tests.

The major benefactions of this work are listed below :

1) Data Secure Storage In order to cover data leakage and rise the difficulty of attack, this paper current a system combining data issuing and data encryption to more data storehouse security.

2) Hierarchical Key Management To cover the key and help the bushwhacker from using the key to recover the data, this paper establish secret allocate and crucial scale derivate algorithm in emulsion with stoner word to enhance crucial security.

3) Experimental Evaluation and Analysis The security check and experimental results show that CSSM can effectively assurance the security of data storehouse, and the rise showing cost is permissible to users.

A short recap of CSSM medium is made in Section 2. Section 3 explains the proposed medium, and Section 4 establish the perpetration of CSSM. The experimental evaluations have been shown in Section 5. We consider several variants and extensions of CSSM in Section 6. Eventually, Section 7 concludes our work.

II. CSSM OVERVIEW

A. REQUIREMENTS ANALYSIS

The main thing of the proposed medium is to strong cloud storehouse opposed to data breach, which may be the end of targeted attack or conduct negligence, in event of hackers indeed some vicious director is suitable to steal stoner data. Aiming at this thing, data collection or encryption is the most generally assume down in multitudinous cases. Both ways could give sequestration- conserve but they also come with endless pitfalls. Data collection spreads data pieces across opposed storehouse areas, but there still lies an occasion to recover data when bushwhackers gain enough pieces. Data encryption technology stores data in cipher textbooks by cracking data with cryptographic keys. Although, bushwhackers can still recover the original data by stealing the keys. That raises the problem of crucial preservation and operation.

Consequently to maximize the confidentiality of cloud data storehouse, the proposed medium should make full use of the superiority of the rule and effectively control its disadvantages. Meantime, the rise cost of the medium should be within a respectablerange.Specifically, following parcels should be met

Property 1 From the standpoint of precluding cloud data confidentiality, any stoner data stored in the cloud would not be declare, viewed, stolen or used by unauthorized existent, Including hackers or some vicious director.

Property 2 On the base of property 1, any parameters like cryptographic keys, which linked to hold on cloud data nonpublic, should also be help.

Property 3 The fresh show outflow of establish proposed medium should be within the stoner's evidence.

ARCHITECTURE OVERVIEW

To recognize main object and properties above, this paper presents CSSM, a cloud secure storage mechanism. Especially the main functions of each layer are as follows:

1) User Layer: This layer is dispose on the user's machine, and the user operates cloud data along the client.

2) Proxy Layer: This layer is dispose in the cloud and produce of proxy nodes with trusted execution environments, like Intel SGX technology and ARM Trust Zone technology .Be sure of execution environment, CSSM programs could execute as excluded. CSSM in proxy layer involves four types: data encryption/decryption, data dispersal, key management and distributed storage.

1 Encryption/Decryption: This type is used to encrypt user uploaded data and decrypt user downloaded data.

2 Data Dispersal: According to the data collect model, the cipher texts is split into several small blocks.

3 Key Management: This type is not only responsible for the generation and maintenance of the key, but also uses the hierarchical key management approach to protect the key.

4 Distributed storage: This module distributes chunked and encrypted data to storage layer.

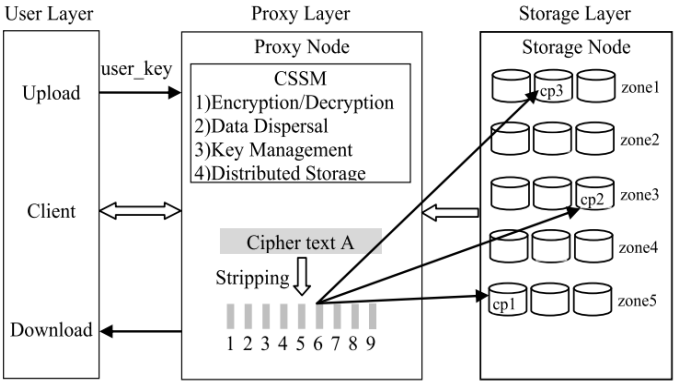
3) Storage Layer: This layer consists of a number of storage nodes that are used to store chunked and encrypted data. Considering data loss or unavailability caused by accident like equipment damage or natural disasters, cloud service providers divide large number of storage nodes into several zones, each of which acts as a failure boundary between multiple copies of the same data.

III. CSSM DESCRIPTION

In the security point of view this paper improves the confidentiality of the user's data in cloud storage, this mechanism ensures the data security and avoid data leakage. The main idea of this mechanism is increasing the security and increasing the hardness of stealing data. At the end, this method process the users data before storing to the cloud storage nodes and though the 2 aspects of work.

CSSM uses data dispersion to split uploaded file into different parts, and each part is called fragment. When users upload files into cloud, files will be splits and stored as fragments into different storage nodes. Compared with undivided way, our mechanism would present to protect data, leading it difficult for attackers to obtain complete data. Alternatively, user files are stored in fragments, it is still possible for attackers to recover the fragments to complete user files as maintained by logical relationship between the contents. As a result, our mechanism establish data encryption. The introduction of data encryption technology is helpful for fragmentation cipher-text storage, and it further rise the difficulty for attackers to steal data.

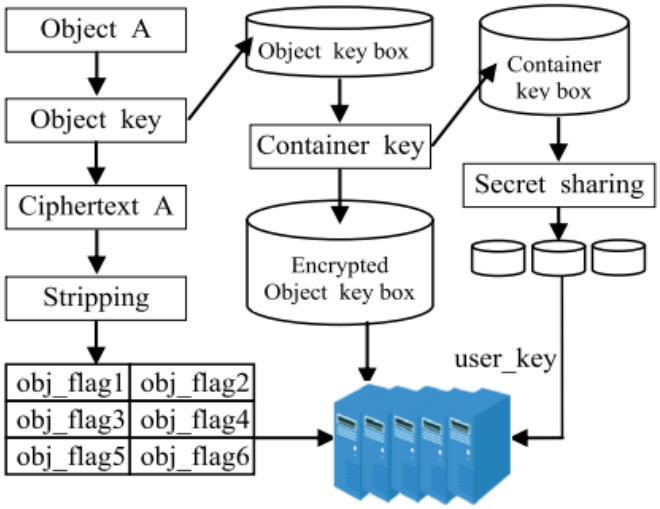
System model architecture



DATA DISPERSION:

In order to stop attackers from stealing all data of the user, CSSM uses the data dispersion method to chunk the data into small parts, and then distribute the parts to different storage nodes. Because of the unpredictability of the location where the user data stored, hackers/attackers could not find location of all parts and recover the files. To further reduce the system overhead, CSSM selected DDS-Striping(1,n,n) as the dispersion model.

However, this method still must recover each parts when the user have to download files. And CSSM must store the record of the relation between user-file and parts. By thefting the record, attackers have the possibility to get the complete user file. Additionally, the code stored within the proxy nodes is also interfere by harmful entities, and therefore the private information in CSSM could also be noticed or hacked during the system operation.



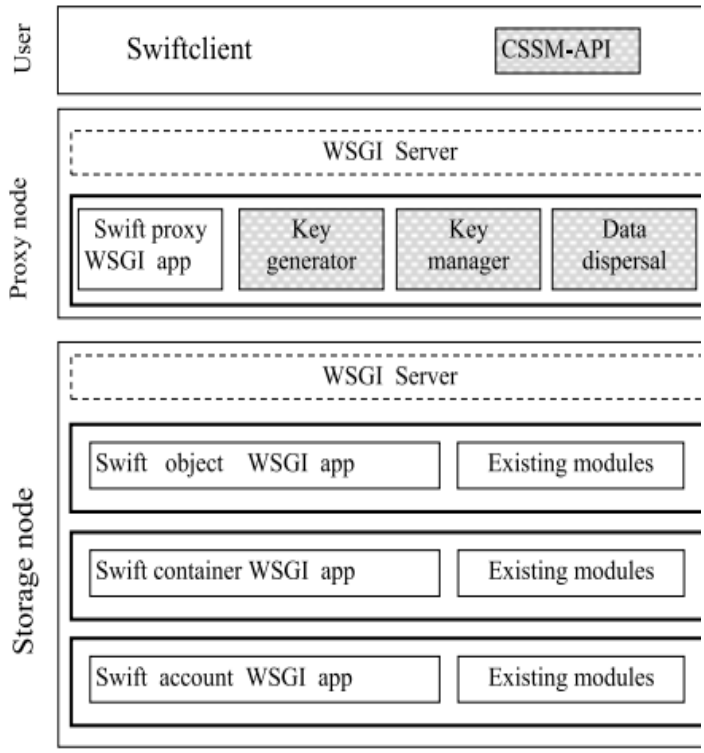
Data Encryption:

Encryption is the most typically used technique to safeguard the user data. In order to decrease time overhead, 128 bit AES symmetric encryption algorithm is used to apply the encryption and decryption for user data. Random key generator should be need in the proxy layer to produce the identical key for encryption and decryption.

Due to the introduction of data encryption, it brings the necessity and importance of key management. We recommend the key hierarchical management method to secure various keys. In cloud storage system, user data is usually as an object and keep in some specific container. In CSSM, every container and object would be allot to a symmetric key, called as object key and container key. Object key is used for encrypting the assigned object.

Apart from that, We create an object key box for every container, in which the keys of all object in the container are stored. After that object key and object box are encrypted by their own container key and saved in cloud storage.

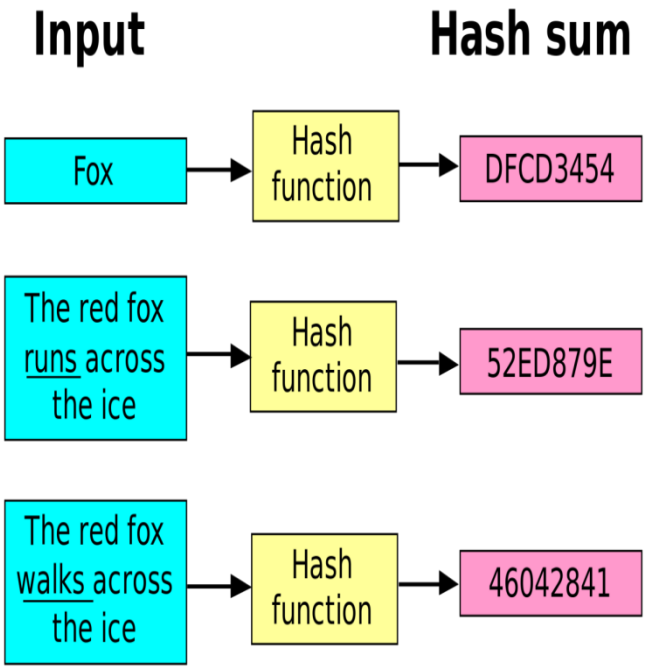
To provide the guarantee to the privacy of the container key, all the user container keys are united to from a container key box. And the secret sharing algorithm used to split container key box to n-block secret sharing blocks and their threshold value is (m,n). To search out these secret sharing blocks, we applying the hierarchical key derivation algorithm to form the index tree whose root node is user_key set by the user. Because the user_key is believe to be a privacy known to the user, it is hard for attackers to acquire the container_key without the user_key, thus make sure the privacy of the container key.



As shown in the diagram, CSSM adopts the "Dispersion and encryption" plan for all objects. 1) Every object is converted into cipher text by encryption, and then divided into various cipher-text parts that would be stored in cloud individually. 2) The object key containing the no of object keys is stored in the cloud after encrypting. 3) By using the hierarchical key derivation algorithm and the secret sharing algorithm, a secret sharing blocks are get generated and get stored in the cloud storage.

Hashing Algorithm :

Hashing algorithms are even as large as encryption algorithms, but some algorithms are commonly used. Some commonly used hashing algorithms are MD5, SHA-1, SHA-2, NTLM, and LANMAN.



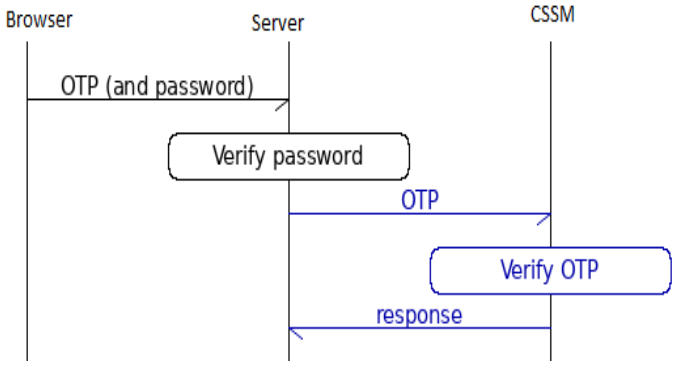
In our system we use SHA-1 algorithm for hashing. This algorithm works by inserting a message as a bit string less than 2^64 bits and produces a hash value of 160-bit hash value known as the message digest.

In our CSSM technique, after the encryption process, we use a hashing algorithm, and hash each part. We provide a separate key to each part and encrypt them. By this process, we can increase the security of user data. Hackers cannot attack the data easily.

So, with this method, we can easily protect the data, and there is no worry about the security of the data. While retrieving the data, it will be decrypt with the key and users can easily download it.

OTP Validation:

An OTP(One Time Password) is a process that generates an alphanumeric/numeric string of characters that authenticates the user for the single session or transaction. This is the most common method that is widely used in many banking / financial sectors for validation. It is mainly used for customer's identity.



In this CSSM technique, when the user wants to use any of the other users' data, we implement OTP method for that validation. When the user wants to read any others' data. We can request the owner of the data by sending them a request.

By this method, the owner can provide access to the user who requests the owner. After the owner grants access, the user can make use of the data. The owner can filter the request and can grant access to the real users. We can easily secure the data and prevent unwanted leakage of data.

This process works on the basis that, when the user wants to use the data / file, they can send a request to the respective owner. The owner of the data can view the request and grant the permission in their own interest.

By this method, the owner can provide access to the user who requests the owner. After the owner grants access, the user can make use of the data. The owner can filter the request and can grant access to the real users. We can easily secure the data and prevent unwanted leakage of data.

This process works on the basis that, when the user wants to use the data / file, they can send a request to the respective owner. The owner of the data can view the request and grant the permission in their own interest.

IV. PROTOTYPE IMPLEMENTATION

Item	Proxy Server	Storage Server
Server Model	PowerEdge R230	System X3650 M3
Numbers of Node	1	4
CPU	8 cores	4 cores
Network Bandwidth	1000Mbps	1000Mbps
Storage Capacity	10TB	292GB per node
Operating System	Ubuntu 14.04 LTS	Ubuntu 14.04 LTS

We apply CSSM prototype based on OpenStack Swift mechanism. The implementation architecture of CSSM. Swift mechanism follows Client/Server architecture. The Swift client contain of a swift client program. We change the swift client program to supply the user interface (CSSM-API) for CSSM. The Swift Server Program assigns the core storage service, which is implemented in the WSGI application. Analysis the design principles of CSSM, we implement data dispersal, key generator and key manager in the WSGI application on Swift proxy node.

When user data needs to be stored, user initiates a storage request containing user key through the CSSM-API interface. The output of the key and encryption operations includes:

- key manager encrypts these generated keys;
- Data dispersal module splits encrypted data.

When user data want to be downloaded, user start a download request containing the user password through the CSSM-API interface. The key generator on proxy node performs the following operations: 1) recover the container key box by using a keyword hierarchy derivation algorithm and user_key; 2) read specific container keys from the container key box; 3) decrypt and read specific object keys from the container key box; 4) perform decryption operations based on the object key, and return the needed data.

V. EXPERIMENTAL ANALYSIS AND EVALUATION

A. EXPERIMENTAL ENVIRONMENT

Each storage server holds two hard disks, one for the system and one for the data. The specific experimental environment.

B. SECURITY ANALYSIS

CSSM enhances the security function of proxy layer, as well as object encryption and dispersion, key generation and management and so on.

No.	0	1	2	3	4	5	6	7	8	9
Content	-^^M--X	^GM-s=e	[^?^X/^	s=eM-VM	u^QM-ko	?^X/^TM	=eM-VMM	/^TM-U0	o^E^YM-	eM-VMM-

As for the container key box, it is processed by the secret sharing algorithm and stored in the cloud in several ‘‘coding blocks’’. Therefore, in order to guarantee the security of user data, it is necessary to prove that the secret sharing algorithm can guarantee the security of these ‘‘encoded blocks’’.The secret sharing algorithm adopts the threshold value[m, n], that is, data D is encoded and converted into n blocks of data, and data D can be recovered at least through m blocks, while any data less than m blocks cannot obtain any part of the metadata information. The proof is given as follows.

1)DISTRIBUTION OF SECRET SHARES

The infinite field GF(q) is used to select n non-similar non-zero elements in the finite field (q is prime, q > n). Note that each element is denoted xi as *xi*xi (*xi* is public). The elements *a*1,*a*2,. . . ,*am*−1 are generated randomly to produce polynomial *f (x)* = *a*0 + *a*1*x*+ . . . +*am*−1*xm*−1 . For the real data D, let D =*a*0 and calculate for *xi*(1 < *i* < *xi*):

The calculated result *f (xi)* is the ‘‘encoded block’’ (1 < i < n).

1. KEY RECOVERY

The recovery process of the original data needs to know at least m blocks of data *f (xi)* in blocks through calculating the following equations(1 < i < m):

$$f(x_1) = a_0 + a_1x_1 + \ldots\ldots + a_{m-1}x_1^{m-1}$$

$$f(x_2) = a_0 + a_1x_2 + \ldots\ldots + a_{m-1}x_2^{m-1}$$

……

$$f(x_m) = a_0 + a_1x_m + \ldots\ldots + a_{m-1}x_m^{m-1}$$

These equations are converted to following matrix:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots\ldots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \ldots\ldots & x_2^{m-1} \\ 1 & x_m & x_m^2 & \ldots\ldots & x_m^{m-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_{m-1} \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_m) \end{bmatrix}$$

$$\text{let } A = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots\ldots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \ldots\ldots & x_2^{m-2} \\ 1 & x_m & x_m^2 & \ldots\ldots & x_m^{m-2} \end{bmatrix}$$

Since A is Vandermonde Matrix, A is invertible and the unique answer could be find in above equations. In different words, we can find the a0,a1,. . . ,am−1, so as to obtain the real data D. But if the no of blocks in f (xi) is lower than m, there’s an infinite no of solutions to (m-1) equations with m unknowns. Therefore, the original data D could not be found.

Hence, when the attacker cannot get the m block ‘‘encoded block’’, the secret sharing algorithm with the threshold value [m,n] can guarantee the security of the container key box. And because of data dispersion in the cloud, the attacker could hardly get m block ‘‘encoded block’’.

C) PERFORMANCE ANALYSIS AND EVALUATION

We analyzed and assessment the performance of CSSM mainly from three aspects: time complexities, space complexities and performance result.

1) TIME COMPLEXITIES ANALYSIS

It encrypts a file of size N in O(N). The keys are stored as object key boxes, each of which is 16 bytes in size. Compared with the size of the user file, the time cost of encryption and decryption of the object key box is very small and can be basically ignored.

Cloud storage secure mechanism adopts keyword hierarchy derivation algorithm and user_key set by user. In this way, a p-layer full binary tree is generated. The number of leaf nodes of the tree is n $=2p-1$, and the total number of nodes is 2n-1. In the implementation, we choose n = 16, so the time cost is small and can be ignored.

2) SPACE COMPLEXITIES ANALYSIS

As result of adopting AES algorithm, the encrypted data is basically the same size as the original data. In terms of keys,each key length is 16 bytes. The storage space of the object key is proportional to the number of user files, and the storage space of the container key is proportional to the number of containers. Relative to the size of user files, all the storage overhead is not large.

3) THE IMPACT OF CSSM ON SYSTEM OVERHEAD

The time overhead of uploading and downloading files with cloud secure storage mechanism rise as the file size rise. By variance, download operations add more runtime overhead than upload operations. For example,for files of 5G size, compared with the rise running time before and after the security enhancement, the file download operation increased by 85 seconds and the file upload operation increased by 78 seconds, when uploading files, the system uses multi-node concurrent operation in the data encryption and dispersion procedure.

4) THE FUNCTIONAL FEATURES OF CSSM

The experimental data show that the time cost of uploading and downloading files increases with the increase of file size, but the growth rate gradually slows down. For the time cost required by files of different sizes, Cloud secure storage mechanism has a better experimental effect for large files and its increase range is low. Therefore, the enhanced security features are more suitable for large files. The experimental results show that CSSM can not only guarantee the confidential storage of data, so as to prevent the leakage of cloud data. And in terms of performance overhead,the increased time overhead is acceptable to the user.

VI. DISCUSSION

In this section we discuss several variants and extensions of Cloud secure storage mechanism which go beyond the scope. In our design, the proxy layer can be integrated into the cloud storage system or its can work as a separate entity. For the sake of quick validation reason, we have used a single proxy server to represent the proxy layer in our implementation. In the experiment, the proxy server we used was equipped with gigabit network card, 8-core CPU (model Xeon E5-2620 V3 2.4GHz or above), 32GB RAM, SAS-300GB hard disk, and 64-bit Ubuntu 14.04 LTS operating system. For the proxy server demands, the general server can meet the demands.

In terms of availability, improvements can be made in the following areas. The current system implementation of Cloud secure storage mechanism is based on a single proxy node, which cannot avoid the problem of single point of failure. If the proxy server crashes, the system will stop running and cloud secure storage mschanism will be unavailable.

To solve this problem, dual-server hot backup or cluster approaches are common solutions we could adopt in the future work. When some servers fail,

the proposed mechanism can still run automatically without interference.

For clusters, the main overhead comes from the cluster construction phase, such as installing service cluster software, adding common data storage devices, and so on. The cluster approach will increase the data synchronization and backup between multiple servers, which in turn improve the availability and efficiency of the cloud storage system.

As for data replication to improve availability, it could be deployed in either proxy layer or cloud storage system. If cloud storage service provider goes for data replication, the cloud system will not only bear data storage overhead, but also consume data replication, network communication overhead. However,for the purpose of not interfering with cloud storage system, almost all data security technologies are implemented in the proxy layer.

Conclusion

For the issue of cloud data disclosure caused by careless management and harmful attack at storage layer, We introduced a new cloud secure storage mechanism. This concept adopted a blend approach of dispersal of data and encryption methods, which can increase the data security and protect data from the attackers. The test results showed that this system can efficiently protect data of the users from leakage. This paper serves a practical approach to clear storage security problem, mainly prevent from data leakage of user data from storage layer. It could also effectively secure cryptographic contents from the storage perspective.

REFERENCE

[1] A. Bhardwaj, F. Al-Turjman, M. Kumar, T. Stephan, and L. Mostarda, ‘‘Capturing-the-invisible (CTI): Behavior-based attacks recognition in IoT-oriented industrial control systems,’’ IEEE Access, vol. 8, pp. 104956–104966, 2020.

[2] M. Kumar, A. Rani, and S. Srivastava, ‘‘Image forensics based on lighting estimation,’’ Int. J. Image Graph., vol. 19, no. 3, Jul. 2019, Art. no. 1950014.

[3] M. Kumar, S. Srivastava, and N. Uddin, ‘‘Image forensic based on lighting estimation,’’ Austral. J. Forensic Sci., vol. 51, no. 3, pp. 243–250, Aug. 2017.

[4] J. Li, Y. Zhang, X. Chen, and Y. Xiang, ‘‘Secure attribute-based data sharing for resource-limited users in cloud computing,’’ Comput. Secur., vol. 72, pp. 1–12, Jan. 2018.

[5] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, ‘‘Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing,’’ Inf. Sci., vol. 379, pp. 42–61, Feb. 2017.

[6] The OpenStack Project. OSSA-2015-006: Unauthorized Delete of Versioned Swift Object. Accessed: Apr. 14, 2015. [Online]. Available: https://security.openstack.org/ossa/OSSA-2015-006.html

[7] The OpenStack Project. OSSA-2015-016: Information Leak Via Swift Tempurls. Accessed: Aug. 26, 2015. [Online]. Available: https://security.openstack.org/ossa/OSSA-2015-016.html

[8] The OpenStack Project. Possible Glance Image Exposure Via Swift. Accessed: Feb. 23, 2015. [Online]. Available: https://wiki.openstack.org/wiki/OSSN/OSSN-0025

[9] Cloud Security Alliance. Top Threats to Cloud Computing: Deep Dive. Accessed: Aug. 8, 2018. [Online]. Available: https://downloads.cloudsecurityalliance.org/assets/research/top-threats/top-threats-to-cloudcomputing-deep-dive.pdf

[10] The OpenStack Project. OpenStack Security Advisories. Accessed: Feb. 2, 2015. [Online]. Available: https://security.openstack.org/ossalist.html

[11] Common Vulnerabilities and Exposures. CVE-2015-5223. Accessed: Jul. 1, 2015. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-5223

[12] Common Vulnerabilities and Exposures. CVE-2016-9590. Accessed: Nov. 23, 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-9590

[13] S. Y. Shah, B. Paulovicks, and P. Zerfos, ‘‘Data-at-rest security for spark,’’ in Proc. IEEE Int. Conf. Big Data (Big Data), Washington DC, USA, Dec. 2016, pp. 1464–1473.

[14] Z. Liu, Y. Huang, J. Li, X. Cheng, and C. Shen, ''DivORAM: Towards a practical oblivious RAM with variable block size,'' Inf. Sci., vol. 447, pp. 1–11, Jun. 2018.
[15] X. Zhang, X. Chen, J. Wang, Z. Zhan, and J. Li, ''Verifiable privacypreserving single-layer perceptron training scheme in cloud computing,'' Soft Comput., vol. 22, no. 23, pp. 7719–7732, Dec. 2018.
[16] C.-Z. Gao, Q. Cheng, P. He, W. Susilo, and J. Li, ''Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack,'' Inf. Sci., vol. 444, pp. 72–88, May 2018.
[17] P. Li, T. Li, H. Ye, J. Li, X. Chen, and Y. Xiang, ''Privacy-preserving machine learning with multiple data providers,'' Future Gener. Comput. Syst., vol. 87, pp. 341–350, Oct. 2018.
[18] B. AlBelooshi, E. Damiani, K. Salah, and T. Martin, ''Securing cryptographic keys in the cloud: A survey,'' IEEE Cloud Comput., vol. 3, no. 4, pp. 42–56, Jul. 2016.
[19] B. AlBelooshi, K. Salah, T. Martin, and E. Damiani, ''Securing cryptographic keys in the IaaS cloud model,'' in Proc. IEEE/ACM 8th Int. Conf. Utility Cloud Comput. (UCC), Limassol, Cyprus, Dec. 2015, pp. 397–401.
[20] P. Zerfos, H. Yeo, B. D. Paulovicks, and V. Sheinin, ''SDFS: Secure distributed file system for data-at-rest security for Hadoop-as-a-service,'' in Proc. IEEE Int. Conf. Big Data, Santa Clara, CA, USA, Oct. 2015, pp. 1262–1271.
[21] J. Zhou, H. Duan, K. Liang, Q. Yan, F. Chen, F. R. Yu, J. Wu, and J. Chen, ''Securing outsourced data in the multi-authority cloud with fine-grained access control and efficient attribute revocation,'' Comput. J., vol. 60, no. 8, pp. 1210–1222, Feb. 2017.
[22] J. Shao, R. Lu, and X. Lin, ''Fine-grained data sharing in cloud computing for mobile devices,'' in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Hong Kong, Apr. 2015, pp. 2677–2685.
[23] S. Han, K. Han, and S. Zhang, ''A data sharing protocol to minimize security and privacy risks of cloud storage in big data era,'' IEEE Access, vol. 7, pp. 60290–60298, 2019.
[24] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie, ''Edge-based differential privacy computing for sensor–cloud systems,'' J. Parallel Distrib. Comput., vol. 136, pp. 75–85, Feb. 2020.
[25] X. Zheng, Y. Zhou, Y. Ye, and F. Li, ''A cloud data deduplication scheme based on certificateless proxy re-encryption,'' J. Syst. Archit., vol. 102, Jan. 2020, Art. no. 101666.