ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 9, Issue 6, June 2022

OPTIMIZING INFORMATION LEAKAGE IN MULTI CLOUD STORAGE SERVICE

[1] R.Kanimozhi ,Assistant Professor,Department of Computer Science and Engineering, Bharathiyar Institute of Engineering for Women

[2]A.Jayasree, P.Kaviya, K.Kayalvizhi, Students in Department of Computer Science and Engineering,Bharathiyar Institute of Engineering for Women

ABSTRACT:**Distributing data over multiple cloud storage providers automatically provides users with a certain degree of information leakage control, for no single point of attack can leak all the information. However, unplanned distribution of data chunks can lead to high information disclosure even while using multiple clouds. In this paper, we study an important information leakage problem caused by unplanned data distribution in multi cloud storage services. Then, we present StoreSim, an information leakage aware storage system in multi cloud. StoreSim aims to store syntactically similar data on the same cloud, thus minimizing the user's information leakage across multiple clouds. We design an approximate algorithm to efficiently generate similarity-preserving signatures for data chunks based on MinHash and Bloom filters, and also design a function to compute the information leakage based on these signatures. Next, we present an effective storage plan generation algorithm based on clustering for distributing data chunks with minimal information leakage across multiple clouds. Finally, we evaluate our scheme using two real datasets from Wikipedia and GitHub. We show that our scheme can reduce the information leakage by up to 60% compared to unplanned placement. Furthermore, our analysis on system attackability demonstrates that our scheme makes attacks on information more complex.**

INDEX TERMS :Cloud computing, data dispersion, data encryption, key management, storage security.

I.INTRODUCTION

Cloud computing has shown remarkable development in recent decades. When stored as a service, it occupies the center stage and backbone for many applications, such as pattern recognition, image forensic [2] and forgery detection [3]. As a result, larger volumes of data will be a part of the cloud area. In the cloud industry, Amazon WebService (AWS) has become the de facto standard. As the core component of the OpenStack that follows this standard,Storage has become one of the most popular cloud storage mechanisms [5].However, Openstack Swift mechanism still faces many real security threats [9]–[11] while providing convenient services. According to Cloud Security Alliance's top threat case analysis report [9] released in 2018, two thirds of the cases will cause user data leakage, mainly due to management negligence and malicious attacks. For instance, under default configuration, OpenStack Swift mechanism typically stores data in plaintext for the sake of performance. That will lead
unauthorized access to user data at the storage layer. In addition, security Report released by Openstack Vulnerability Management Team VMT, the Swift mechanism may leak user data or configuration information in virtue of security vulnerabilities [11], [12].

Shah *et al.* [13] proposed a cloud-oriented data security storage mechanism under the framework of Apache Spark, which prevents data leakage and improves the security of the Apache Spark framework. To protect user data on the cloud, different encryption schemes [14]–[17] have been adopted to avoid information leakage during the machine learning process. Nevertheless, above research requires secure key management mechanisms to prevent cryptographic material exposure [18], [19].

Zerfos *et al.* [20] constructed a secure distributed storage system based on Hadoop system, which keep the confidant tiality of cloud data through data dispersion and encryption. It performs the data decryption and assembly tasks before reading data. To prevent the keys from being stolen, this method requires key cache server and all keys should be stored in memory only. Some approaches [21], [22] intro duced independent third party to manage the key. It is assumed that third parties stay trusted. However, the assumption cloud not always exists in the real cloud storage environments [23].

Wang *et al.* [24] presented a data privacy preserving scheme for sensor-cloud system, based on edge computing

and differential storage method. In this scheme, user data would be divided into different parts and stored in local, edge and cloud layers respectively. But the scheme relies on the characteristics of data from wireless sensor networks, and requires skilled users to manage the edge servers. To improve the efficiency and decrease the redundancy, Zheng *et al.* [25]provided a cloud data deduplication scheme to detect and remove identical user data in the cloud. However, from the perspective of preventing data loss due to disaster, a certain number of copies should be sent to multiple regions.

In a word, to protect cloud data from leakage at the storage layer, this paper presents CSSM, a Cloud Secure Storage Mechanism. CSSM combines data dispersion with data encryption, so that large-scale cloud data and keys would be stored in chunked cipher texts. On this basis, user password and secret sharing are introduced to further protect key security. We implemented CSSM based on OpenStack Swift Mechanism and made several tests.

The major contributions of this work are listed below:

1) *Data Secure Storage:* In order to prevent data leakage and increase the difficulty of attack, this paper presents a method combining data distribution and data encryption to improve data storage security.

2) *Hierarchical Key Management:* To protect the key and prevent the attacker from using the key to recover the data,This paper introduces secret sharing and key hierarchy derivation algorithms in combination with user password to enhance key security.

3) *Experimental Evaluation and Analysis:* The security analysis and experimental results show that CSSM can effectively guarantee the security of data storage, and the increased performance cost is acceptable to users Remainder of the paper is organized as follows:A brief overview of CSSM mechanism is made inSection 2. Section 3 explains the proposed mechanism, andSection 4 introduces the implementation of CSSM. The Experimental evaluations have been shown in Section 5.We discussed several variants and extensions of CSSM inSection 6. Finally, Section 7 concludes our work.

II. CSSM OVERVIEW

A. REQUIREMENTS ANALYSIS

The main objective of the proposed mechanism is to secure cloud storage against data breach, which may be the result of targeted attack (e.g. disk cloning) or management negligence(e.g. misconfiguration), in case hackers or even some malicious administrator is able to steal user data.

Aiming at this goal, data dispersion or encryption is the most commonly adopted way in numerous cases. Both techniques could provide privacy-preserving, but they also come with inherent risks. Data dispersion spreads data pieces across different storage areas, but there still lies an opportunity to recover data when attackers obtain enough pieces. Data encryption technology stores data in cipher texts by encrypting data with cryptographic keys. However, attackers can still recover the original data by stealing the keys. That raises the problem of key protection and management. Therefore,to maximize the confidentiality of cloud data storage, the proposed mechanism should make full use of the advantages of the method and effectively control its disadvantages. Meanwhile, the increased cost of the mechanism should be within a reasonable range.Specifically, following properties should be met:

Property 1: From the perspective of protecting cloud data confidentiality, any user data stored in the cloud would not be released, viewed, stolen or used by unauthorized individuals,such as hackers or some malicious administrator.

Property 2: On the basis of property 1, any parameters like cryptographic keys, which are related to keep cloud data confidential, should also be protected.

Property 3: The additional performance overhead of deploying proposed mechanism should be within the user's acceptance.

B. ARCHITECTURE OVERVIEW

To realize primary object and properties above, this paper presents CSSM, a cloud secure storage mechanism. As shown,in Figure 1, CSSM could be divided into three layers: Theuser layer, the proxy layer, and the storage layer. Specifically,the main functions of each layer are as follows:

1) *User Layer:* This layer is deployed on the user's machine, and the user operates (upload, download, etc.) cloud data through the client.

2) *Proxy Layer:* This layer is deployed in the cloud and composed of proxy nodes with trusted execution environments, such as Intel SGX technology [26] and ARM Trust Zone technology [27]. In trusted execution environment, CSSM programs could perform as expected. CSSM in the proxy layer includes four modules: data encryption/decryption, data dispersal, key management and distributed storage.

Ⓒ *Encryption/Decryption:* This module is used to encrypt user uploaded data and decrypt user downloaded data.

Ⓒ *Data Dispersal:* According to the data dispersal model, The cipher text is divided into several small blocks.

Ⓒ*Key Management:* This module 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 the 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

From the perspective of improving the confidentiality of user data in the cloud, this paper presents CSSM, a cloud secure
mechanism to ensure data security and avoid data breach. The Core idea of the CSSM mechanism is to increase the difficulty of stealing data. To this end, CSSM process data before storing to cloud storage nodes through two aspects of work.

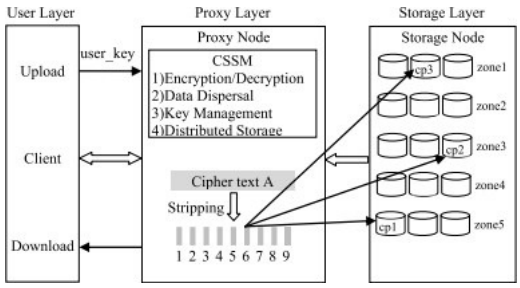

Figure1:CSSM system model architecture

First, CSSM uses data dispersion to divide uploaded files into several parts, and each part is called a fragment. When users upload files into the cloud, files will be divided and storedas fragments into different storage nodes. Compared with an undivided way, our mechanism would contribute to protectdata, making it difficult for attackers to obtain complete data.On the other hand, although user files are stored in fragments, it is still possible for attackers to recover the fragments to complete user files according to the logical relationship between the contents. Therefore, our mechanism introduces data encryption. The introduction of data encryption technology is helpful for fragmentation cipher-text storage, and it further increases the difficulty for attackers to steal data.

A. DATA DISPERSION

In order to prevent attackers from stealing complete user data,CSSM first uses data dispersion technology to split data into fragments, and then distribute the fragments to different storage nodes. Due to the uncertainty of the location where data was stored, attackers could hardly locate all fragments and recover user files. To further reduce system overhead, CSSM selected DDS——Stripping(1,n,n) [28] as the dispersal model.However, cloud storage mechanism still needs to restore each fragment when the user downloads fifiles. And CSS needs to keep the record of the composition relationship between user file and fragment. By stealing the record, attackers have the opportunity to obtain the complete user file.

In addition, the code stored in the proxy nodes may be viewed or tampered with by malicious entities, and the sensitive information in CSSM may be maliciously spied on or stolen during the operation of the system.In response to the above security threats, we believe those can be dealt with by trusted computing technology (such as trusted execution environment and remote attestation technology). The problems that trusted execution environment
addresses can be twofold. One is to prevent the code of the proxy service node from being tampered, which usually adopts integrity measurement and remote attestation technology to determine whether the code should be trusted.

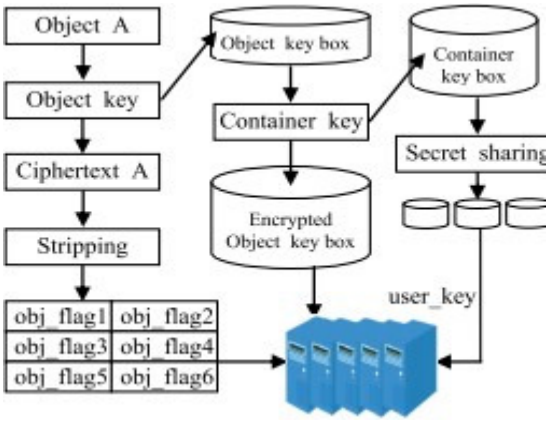

FIGURE 2. CSSM system mechanism

On the other hand, trusted execution environment could ensure that CSSM code runs without interference. We assumeCSSM is implemented in a trusted execution environment on proxy nodes. Detection mechanisms and monitoring mechanisms are needed to prevent code tampering and ensure the trusted execution of programs in case of malicious behavior.Nowadays there are many achievements that can be used for reference [29], [30]. The CSSM mechanism is based on the research of trusted computing.

B. DATA ENCRYPTION

Encryption is the most common way to secure user data. In order to reduce time overhead, 128 bit AES symmetric an encryption algorithm is selected to implement the encryption and decryption for user data. Key generator should be required in the proxy layer to generate the symmetric key for encryption and decryption. Due to the introduction of data encryption, it brings the necessity and importance of key management. As shown In Figure 2, we put forward the key hierarchical management method to

protect various keys. In cloud storage system, user data is usually regarded as an object and stored in some

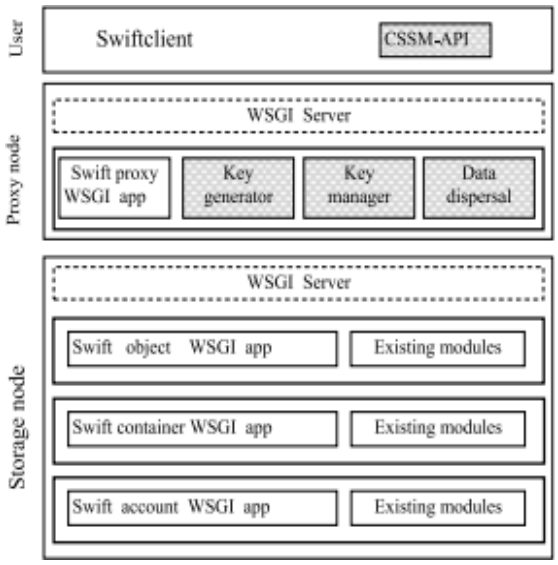

FIGURE 3. CSSM system implementation architecture.

specified container. In CSSM, each object and container would be assigned a symmetric key, respectively called object key and container key. Object key would be used for encrypting assigned objects. That is to say, the confidential issue of objects will be turned to the same number of object keys.Besides, we design an object key box for each container,in which the object keys of all objects in the container are stored. Then, all the object keys in the object key box are encrypted by their container key and stored in the cloud. Similarly, the confidential issue of object keys will be changed to fewer containersIn order to guarantee the security of the container key,all the container keys of a user are integrated to form a container key box. A secret sharing algorithm [31] is used to divide a container key box into n-block secret sharing blocks whose threshold value is (m, n). To locate these secret sharing blocks, we use the hierarchical key derivation algorithm [32]to establish the index tree whose root node is user_key set by the user. Because the user_key is considered to be a secret only known to the user, it is difficult for attackers to obtain the container key without the user_key, thus ensuring the confidentiality of the container key. In addition, we regard
the problem of updating the index tree caused by the change of user_key as a problem to be solved in the future.

As shown in Figure 2, CSSM adopts the ‘‘dispersion and encryption’’ strategy for all objects: 1) each object is
encrypted to cipher-text first, and then divided into several cipher-text fragments that would be stored in the cloud sepa rarely; 2) the object key box consisting of a number of object keys is encrypted by the container key and stored in the cloud;

3) by secret sharing and hierarchical key derivation algorithm,secret sharing blocks are generated and stored in the cloud.

IV. PROTOTYPE IMPLEMENTATION

We implemented a CSSM prototype based on the OpenStack Swift mechanism. The implementation architecture of CSSM is shown in Figure 3. Swift mechanism follows Client/Server

TABLE 1. Prototype system experimental environment

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

architecture. The Swift client consists of a swiftclient pro gram. We modify the swiftclient program to provide the user interface (CSSM-API) for CSSM.

In addition, the Swift Server Program provides the core storage service, which is implemented in the WSGI applica tion. Considering the design principles of CSSM, we imple ment data dispersal, key generator and key manager in the WSGI application on Swift proxy node.

[10] discussed that Helpful correspondence is developing as a standout amongst the most encouraging procedures in remote systems by reason of giving spatial differing qualities pick up. The transfer hub (RN) assumes a key part in agreeable correspondences, and RN choice may generously influence the execution pick up in a system with helpful media get to control (MAC).

When user data needs to be downloaded, user initiates a download request containing the user password through the CSSM-API interface. The key generator on proxy node per forms the following operations: 1) recover the container key box by using a keyword hierarchy derivation algorithm anduser_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

The CSSM prototype consists of five servers, one as the proxy service node and the other four as the storage service node.Each storage server holds two hard disks, one for the system and one for the data. The specific experimental environment is shown in Table 1.

B. SECURITY ANALYSIS

CSSM enhances the security function of the proxy layer, including object encryption and dispersion, key generation and management and so on. We created a container called ‘‘encon’’ for storing encrypted data, tested files called ‘‘ftxt’’and ‘‘ukey’’ as our user password. The operation steps are as follows:

~# cat fttx I wrote this vfile to do a test. I want to read the file from the storage device directly. Can I be successful?. . . . . .

. . . . . .

~#swift −V3 post encon −u ukeys

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-U8	o^E^YM-	eM-VMM-

~#swift −V3 upload encon ftxt −u ukey

~#swift −V3 download encon ftxt −u ukey

As shown in figure 4, ifile ‘‘text’’ has been divided into 10 blocks, and each block is stored in ciphertext form. In addition, the object key box is also stored in cipher text. 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 userdata, 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 finite fifield GF(q) is used to select n different non-zero elements in the fifinite fifield (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 form a polynomial $f(x) = a0 + a1x + + am-1xm-1$. For the original data D, let D =*a*0 and calculate for $xi(1 < i < xi)$:$f(x) = a0 + X\ mj = -11$

aj

J

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

2) KEY RECOVERY

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

$f(x1) = a0 + a1x1 + \quad + am-1xm$

−1

1

$f(x2) = a0 + a1x2 + \ldots + am-1x2m-1$

. . .

$f(xm) = a0 + a1xm + \ldots + am-1xmm-1$

These equations are converted to following matrix:

Since A is Vandermonde Matrix, A is invertible and the a unique solution could be found in above equations. In other words, we can find the *a*0,*a*1,. . . ,*am*−1, so as to obtain the original data D. But if the number of blocks in *f (xi)* is less than m, there’s an infinite number of solutions to (m-1)equations with m unknowns. Therefore, the original data Could not be obtained.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 my block ‘‘encoded block’’.

C. PERFORMANCE ANALYSIS AND EVALUATION

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

1) TIME COMPLEXITIES ANALYSIS

CSSM uses 128-bit AES encryption, so the time cost of encryption is proportional to the size of the encrypted file. And 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. The size of the object key box is proportional to the number of user files in the container. Compared with the size of the user file, the time cost of encryption and decryption of the object The key box is very small and can be basically ignored. As for the index of keys, CSSM adopts keyword hierarchy derivation algorithm and user_key set by user. In this way , aa p-layer full binary tree is generated. The number of leaf nodes of the tree is n =2*p*−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 storage overhead is not large.

3) PERFORMANCE RESULTS

In order to evaluate the performance of CSSM, we compare the time overhead with and without using CSS in upload and download operations based on the Swift system.In general, normal file sizes range from

32KB to 5GB.Considering the randomness of the time to complete each

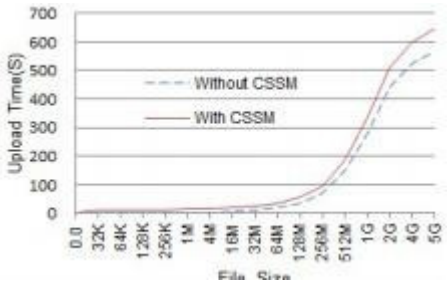

FIGURE 5 File upload time cost comparison.

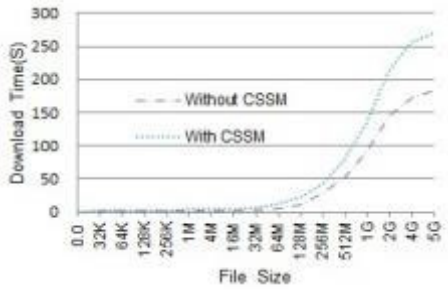

FIGURE 6 File download time cost comparison.

operation, we do 10 experiments on the same data file upload and download respectively, and the time consumed is average.As shown in figure 5 and figure 6, additional time overhead spent on upload and download operations are illustrated respectively. By analyzing the experimental results, we can draw the following conclusions.

4) THE IMPACT OF CSSM ON SYSTEM OVERHEAD

According to the experimental results, the time overhead of uploading and downloading files with CSSM increases as the file size increases. [8] discussed because of various appealing focal points, agreeable correspondences have been broadly viewed as one of the promising systems to enhance throughput and scope execution in remote interchanges.

The hand-off hub (RN) assumes a key part in helpful interchanges, and RN determination may considerably influence the execution pick up in a system with agreeable media get to control (MAC). When downloading a file,the system needs to locate the location of each cipher-text fragment, and then recover from the fragments, which is a serial operation.

5) THE FUNCTIONAL FEATURES OF CSSM

The experimental data show that the time cost of uploading and downloading files increases with the increase of filesize, but the growth rate gradually slows down. For the time cost required by fifiles of different sizes, CSSM has a better experimental effect for large files and its increased 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 confidentiality 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.

IV. DISCUSSION

In this section we discuss several variants and extensions of CSSM which goes beyond the scope. From the perspective of improving storage security, a secure mechanism based on the proxy layer is proposed. In our design, the proxy layer can be integrated into the cloud storage system or it 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 a 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 requirements, the general server can meet the requirements.In terms of availability, improvements can be made in the following areas. 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. The proxy layer is designed to enhance the security of cloud storage, so it covers a number of data security technologies such as data encryption and dispersion. As for data replication to improve availability, it could be deployed in either proxy layer or cloud storage system. If a 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 systems, almost all data security technologies are implemented in the proxy layer. [4] proposed a secure hash message authentication code. A secure hash message authentication code to avoid certificate revocation list checking is proposed for vehicular ad hoc networks (VANETs). [6] discussed that the activity related status data will be communicated consistently and shared among drivers through VANETs keeping in mind the end goal to enhance driving security and solace. Along these lines, Vehicular specially appointed systems (VANETs) require safeguarding and secure information correspondences.

VII. CONCLUSION

For the issue of cloud data leakage caused by management negligence and malicious attack at the storage layer, we proposed CSSM, a cloud secure storage mechanism. CSSM adopted a combined approach of data dispersal and encryption 63750 VOLUME 9, 2021H. Song *et al.*: CSSM Based on Data Dispersion and Encryption technologies, which can improve the data security and prevent attackers from stealing user data. The experimental results show that CSSM can effectively prevent user data leakage at the cloud

storage layer. In terms of performance, the increased time overhead of CSSM is acceptable to users. This paper provides a feasible approach to solve the storage security problem, especially prevention from user data leakage at cloud storage layer. CSSM could also effectively protect cryptographic materials from storage perspective.

REFERENCES

[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] Christo Ananth, M.Danya Priyadharshini, "A Secure Hash Message Authentication Code to avoid Certificate Revocation list Checking in Vehicular Adhoc networks", International Journal of Applied Engineering Research (IJAER), Volume 10, Special Issue 2, 2015,(1250-1254).

[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] Christo Ananth, Dr.S. Selvakani, K. Vasumathi, "An Efficient Privacy Preservation in Vehicular Communications Using EC-Based Chameleon Hashing", Journal of Advanced Research in Dynamical and Control Systems, 15-Special Issue, December 2017,pp: 787-792

[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] Christo Ananth, Dr. G. Arul Dalton, Dr.S.Selvakani, "An Efficient Cooperative Media Access Control Based Relay Node Selection In Wireless Networks", International Journal of Pure and Applied Mathematics, Volume 118, No. 5, 2018,(659-668)

[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-cloud computing-deep-dive.pdf

[10] Christo Ananth, Joy Winston.J., "SPLITTING ALGORITHM BASED RELAY NODE SELEC-TION IN WIRELESS NETWORKS", Revista de la Facultad de Agronomía, Volume 34, No. 1, 2018,(162-169)

[11] [11] CommonVulnerabilities and Exposures. *CVE-2015-5223*. Accessed:Jul.1,201https://cve.mitre.org/cgi-bin/cvename5.[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]. AvailableHttps://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 privacy preserving 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. AlBalooshi, 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. AlBalooshi, 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.