IMPLEMENTING MULTI-FACTOR AUTHENTICATION FOR ENHANCED CYBER SECURITY IN CLOUD STORAGE

Abstract

Cloud storage service has shown its extraordinary power and wide prominence which offers key help for fast improvement of distributed computing. However, there are still significant security incidents that result in the leakage of large amounts of sensitive data at the cloud storage layer as a result of management carelessness and malicious attack. According to the viewpoint of safeguarding cloud information privacy, we proposed a Cloud Secure Capacity System named CSSM.

CSSM implemented encrypted, chucked, and distributed storage by integrating data dispersion and distributed storage in order to prevent data breaches at the storage layer. In addition, CSSM employed a hierarchical management strategy and combined secret sharing with user passwords to stop the leakage of cryptographic materials. The exploratory outcomes suggest that the task is not only suitable for ensuring data security at the storage layer to prevent leakage but also efficient in storing a large amount of cloud data without causing significant delays. For instance, when users upload or download a 5GB file using CSSM, it only takes approximately 646 seconds for uploading and 269 seconds for downloading. These processing times are considered acceptable and should not cause any inconvenience for the users.

Overall, the findings indicate that the proposed approach using CSSM is effective in achieving both data security and efficient cloud data storage.

Keywords: Authentication, Cyber Security, Cloud Storage.

Author

Dr. Prathibha G
Associate Professor
Information Science and Engineering
Rajeev Institute of Technology
Hassan
prathibhaghebbar@rithassan.ac.in

I. INTRODUCTION

Based on the provided text, the major contributions of the work on the Cloud Secure Storage System (CSSM) are as follows:

1. **Data Security:** CSSM is designed to safeguard cloud data from leakage at the storage layer. It achieves this through a combination of data scattering and data encryption. Data and keys are stored in fragmented ciphertext, making it difficult for unauthorized parties to access sensitive information.

2. **Integration with OpenStack Swift:** CSSM is implemented based on the OpenStack Swift system, which is a popular distributed storage platform. This integration allows for seamless incorporation of the security mechanisms into the existing infrastructure.

3. **Key Management:** The paper addresses the challenge of secure key management to prevent cryptographic materials from being exposed. It introduces techniques like user password protection and secret sharing to enhance the security of keys.

4. **Efficiency and Deduplication:** The work aims to improve efficiency and reduce redundancy in cloud data storage. It presents a cloud data duplication scheme to identify and eliminate duplicate client data in the cloud, helping to optimize storage utilization.

5. **Disaster Recovery:** Considering the prevention of data loss due to disasters, the paper discusses the need to send a certain number of data copies to different locations, enhancing data resilience.

6. **Edge Computing Integration:** The proposed system takes into account edge computing and differential storage to ensure data privacy in the sensor-cloud system.

7. **Real-world Relevance:** The paper acknowledges that the assumption of a trustworthy third party in cloud storage environments may not always hold, making the design of CSSM more practical in real-world scenarios.

Overall, the Cloud Secure Storage System (CSSM) presents a comprehensive approach to enhance the security of cloud data at the storage layer, considering various aspects such as encryption, key management, data duplication, and edge computing. The system is implemented and tested on the OpenStack Swift platform, demonstrating its feasibility and potential in real-world cloud storage environments.

II. METHODOLOGY

1. **CSSM (Cloud Secure Storage Mechanism):** The proposed system aims to provide distributed storage that is resilient against data breaches resulting from various threats, including targeted attacks (e.g., disk cloning) or management negligence (e.g., misconfiguration). The primary objective is to protect user data even in the event of malicious actions by hackers or unauthorized access by administrators. To achieve this goal, the paper considers two main approaches: data dispersion and data encryption. Both methods aim to enhance security, but they also come with their own inherent risks.

Data dispersion involves distributing data fragments across different storage locations, making it more challenging for attackers to reconstruct the complete data set. However, there is still a potential risk that attackers could retrieve enough fragments to recover the original data.

On the other hand, data encryption involves encoding data with cryptographic keys, converting it into ciphertext, and storing it securely. Encryption provides strong protection against unauthorized access, but it relies heavily on the robustness of the encryption algorithm and the secure management of cryptographic keys.

The paper suggests that a combination of these two approaches, data dispersion, and data encryption, can be utilized to enhance data security further. By employing both techniques, the system can provide a higher level of protection against data breaches, even if attackers manage to obtain some fragments or gain access to the encrypted data. This approach aims to mitigate the risks associated with individual methods and improve the overall security of the distributed storage system.

2. **Data Encryption:** Correct, data encryption is a crucial technique used to protect sensitive information and ensure data security. It involves converting data into a coded form called ciphertext, which can only be deciphered back into its original plaintext form by authorized parties who possess the secret key or password required for decryption.

   There are two main types of data encryption:

   - **Symmetric Encryption:** In symmetric encryption, a single secret key is utilized both for encryption and decoding. This implies that a similar key is utilized to encode information at the sender's end and decode it at the receiver's end. While this method is efficient and fast, the challenge lies in securely sharing the secret key with authorized parties without it falling into the hands of unauthorized users.

   - **Asymmetric Encryption (Public-key Encryption):** Asymmetric encryption utilizes a couple of keys, a public key, and a private key. The public key is utilized for encryption, and anyone can have access to it. However, just the recipient has the corresponding private key to decrypt the data. This method addresses the key distribution issue of symmetric encryption and allows secure communication between parties without sharing the private key.

     Both symmetric and asymmetric encryption have their use cases and advantages, and they are often used together in a hybrid approach to provide a robust data security solution. Data encryption plays a significant role in securing sensitive data during transmission and storage, and it is widely adopted by businesses, governments, and individuals to protect their valuable information from unauthorized access and potential data breaches.

3. **Purpose:** Absolutely, the primary purpose of data encryption is to protect the privacy and security of digital information while it is stored on computer systems or transmitted over

the internet or other computer networks. Encryption ensures that sensitive data remains confidential and cannot be accessed or understood by unauthorized individuals.

Older encryption standards like the Data Encryption Standard (DES) have been supplanted by more modern and robust encryption algorithms. These modern algorithms play a crucial role in ensuring the security of IT systems and communications.

Encryption algorithms provide several key security features, including:

- **Authentication:** Encryption considers the confirmation of a message's origin. This ensures that the sender's identity can be confirmed, helping to prevent unauthorized users from masquerading as legitimate senders.

- **Integrity:** Encryption provides the means to verify that a message's contents have not been altered or tampered with during transmission. This ensures that the information remains intact and unchanged, maintaining data integrity.

- **Non-repudiation:** Encryption guarantees that a message sender cannot deny having sent a particular message. It prevents the sender from disowning their actions and provides proof of their involvement.

By incorporating encryption into IT systems and communication protocols, organizations and individuals can significantly enhance their data security and protect sensitive information from unauthorized access and potential data breaches. It is an essential tool for maintaining confidentiality, integrity, and authenticity in the digital world.

III. DATA DECRYPTION

Decoding is the most common way of changing information that has been delivered incomprehensible through encryption back to its decoded structure. The system decrypts the muddled data by extracting it, transforming it into text and images that the reader and the system can easily comprehend. Unscrambling might be achieved physically or consequently. Additionally, it can be carried out using a set of keys or passwords. Privacy is one of the main reasons to use an encryption-decryption system. Information can be viewed and accessed by unauthorized individuals or organizations as it travels over the World Wide Web. Accordingly, information is encoded to lessen information misfortune and burglary. E-mail messages, text files, images, user data, and directories are all common examples of items that can be encrypted. The individual responsible for decoding gets a brief or window where a secret phrase might be placed to get to encoded data.

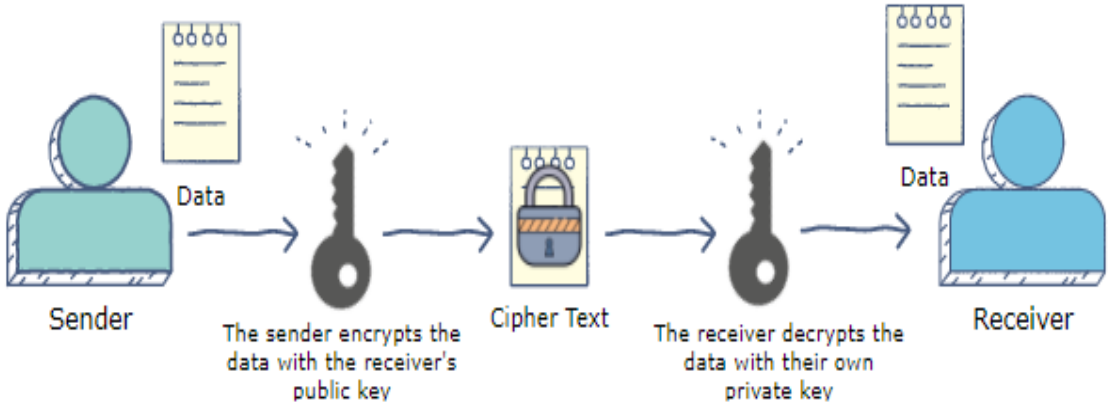


Figure 1: Data Encryption and Decryption

IV. ARCHITECTURE

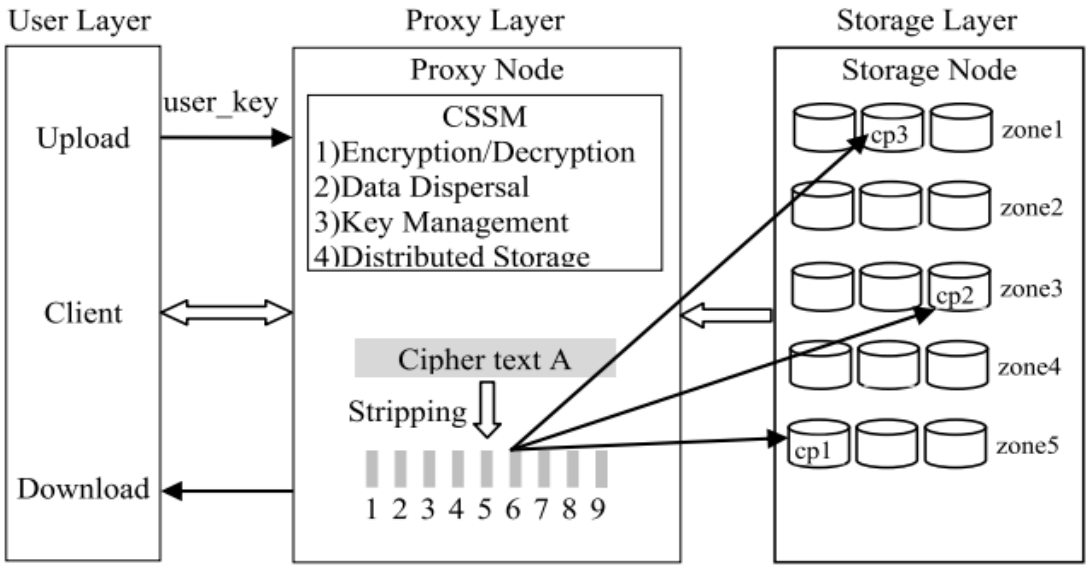


Figure 2: Architecture

To acknowledge essential item and properties over, this paper presents CSSM, a cloud secure capacity component. CSSM could be divided into three layers, as depicted in Figure: The client layer, the intermediary layer, and the stockpiling layer. In particular, the primary elements of each layer are as per the following:

1. **User Layer:** This layer is conveyed on the client's machine, and the client works (transfer, download, and so forth.) through the client, cloud data.

2. **Proxy Layer:** This layer is conveyed in the cloud and made out of intermediary hubs with confided in execution conditions, for example, Intel SGX innovation and ARM Trust

Zone innovation. CSSM programs could carry out as expected in an environment that was trusted for execution. CSSM in intermediary layer incorporates four modules: information encryption/decoding, data dispersal, key management and distributed storage.

The provided information outlines the main modules of a system designed to ensure secure data storage in a distributed environment. Let's briefly describe each module:

- **Encryption/Decryption:** This module handles the encryption of user-uploaded data, converting it into ciphertext using encryption algorithms and a secret key. On the other hand, it also performs the decryption process when users download data, converting ciphertext back into its original plaintext form using the appropriate decryption key.

- **Data Dispersal:** According to the data dispersal model, the ciphertext is divided into smaller blocks or chunks. This fragmentation ensures that data is distributed across multiple storage locations, enhancing security by making it more challenging for attackers to recover the entire data set even if they gain access to some parts of it.

- **Key Management:** This module is responsible for key generation, distribution, and maintenance. It plays a critical role in securing the cryptographic keys used for encryption and decryption. Additionally, it employs a hierarchical key management approach to enhance key security. Hierarchical key management typically involves creating a tree-like structure of keys, where higher-level keys protect lower-level keys, and access is controlled to ensure authorized users have access to the appropriate keys.

- **Distributed Storage:** This module handles the storage and distribution of the chunked and encrypted data to various storage locations within the distributed environment. The distribution ensures redundancy and fault tolerance, improving the system's robustness against data loss or hardware failures.

Overall, this system combines encryption, data dispersal, key management, and distributed storage to provide a comprehensive solution for securely storing data in a distributed environment. The combination of these modules helps ensure data confidentiality, integrity, and availability, protecting the data from unauthorized access and potential security threats.

3. **Storage Layer:** A number of storage nodes are used to store encrypted and chunked data in this layer. Taking into account information misfortune or inaccessibility brought about coincidentally like hardware harm or catastrophic events, cloud specialist co-ops partition enormous number of stockpiling hubs into a few zones, every one of which goes about as a disappointment limit between numerous duplicates of similar information.

V. SYSTEM DESIGN

1. **Proxy:** Proxy can login with valid credentials; proxy is responsible to add Sender/Receivers Information and send login information to the sender/receiver through email. Proxy also view requests from the receiver

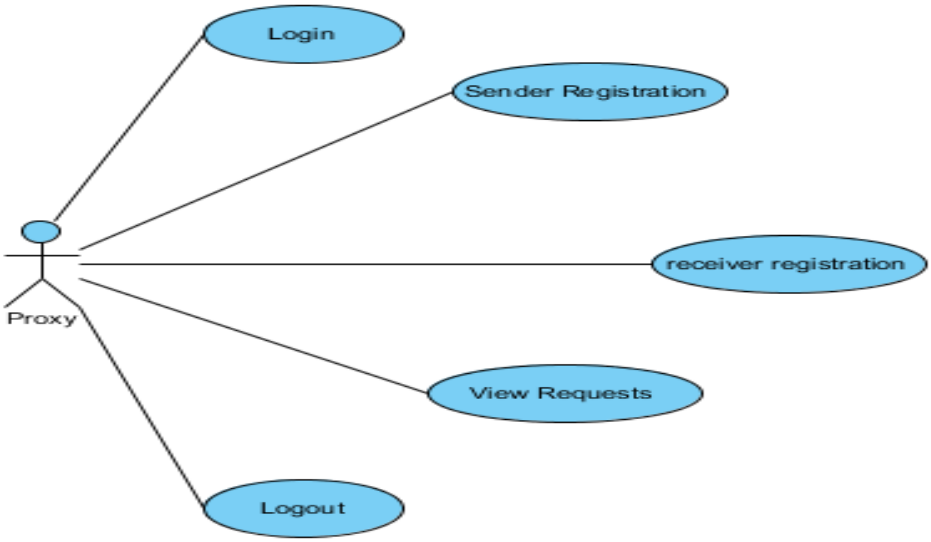


Figure 3: Proxy

2. **Sender:** Sender will login with the credentials which are sent by the proxy sender is responsible to upload files, encrypt files and to generate trapdoor view Files and Delete those files.

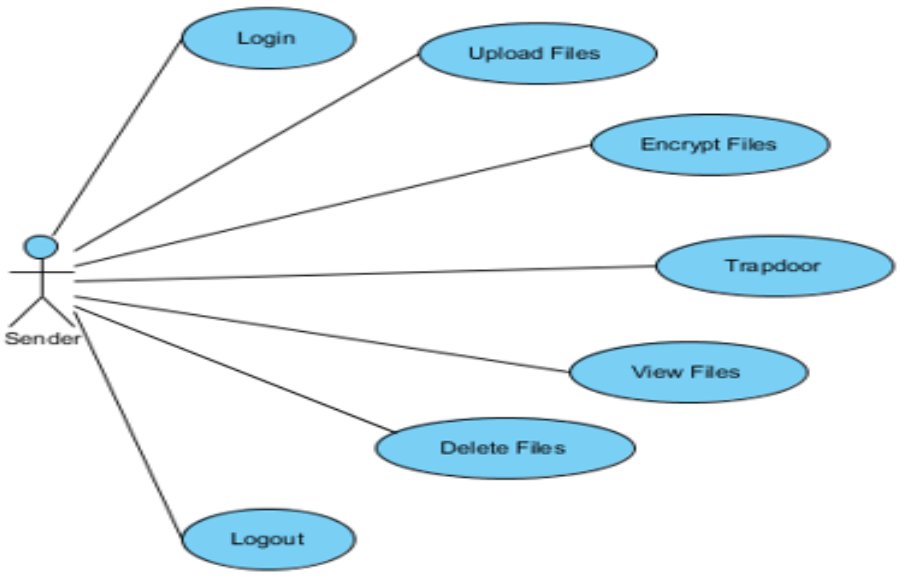


Figure 4: Sender

3. **Receiver:** Receiver also login as Sender but performs different operations like view all files which are uploaded by the sender and send's request to the proxy to view the file data and view files which are accepted by the proxy. And finally, logout from the site.

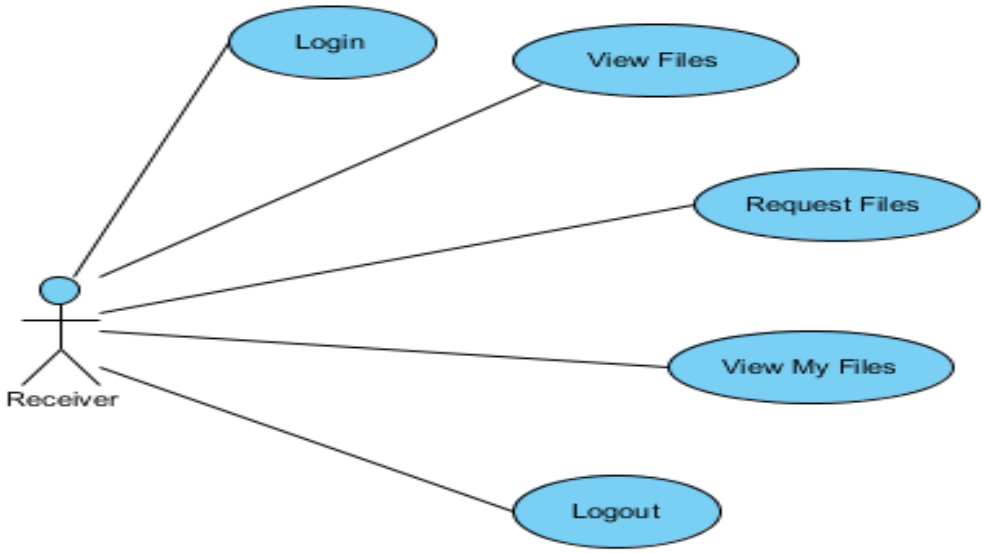


Figure 5: Receiver

4. **Cloud Service Provider:** Cloud service provider login with valid credentials and view all the server's data and verifies files are attacked by the attacker and provide some security to the files and logout from the system.

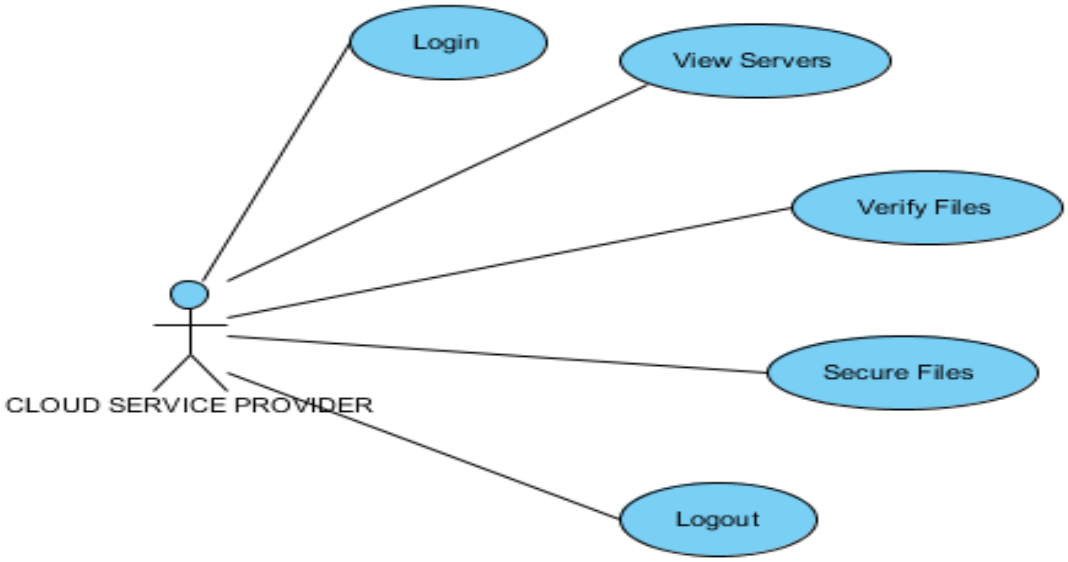


Figure 6: Cloud Service Provider

5. **Attacker:** Attacker login with malicious content and attacks the files to destroy or to theft files. Logout from the site.

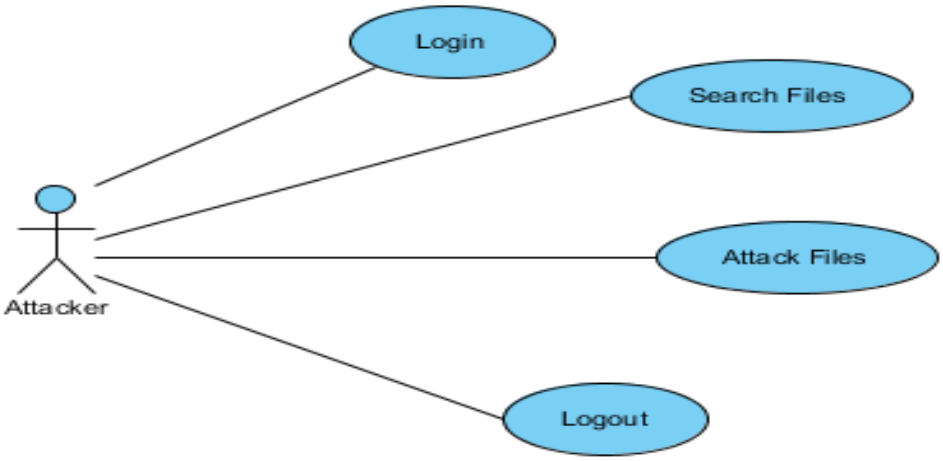


Figure 7: Attacker

VI. DEPLOYMENT DIAGRAM

A deployment diagram is one of the UML (Unified Modeling Language) diagrams that represent the deployment view of a system. It shows how the components of a software system are deployed on hardware nodes or computing devices.
In a deployment diagram:

1. **Components:** The components refer to the software modules or building blocks that make up the system. These components are typically represented as boxes in the diagram and may include executable files, libraries, databases, etc.

2. **Nodes:** Nodes represent the physical hardware or computing devices on which the components are deployed. These can be servers, personal computers, mobile devices, routers, or any other physical device that runs the software.

3. **Deployment Relationships:** The deployment diagram illustrates how the components are connected to the nodes. This connection indicates the deployment of components on specific hardware. For example, you might have a web server component deployed on a particular server node, a database component deployed on a separate database server node, etc.

4. **Communication Paths:** Deployment diagrams may also show communication paths between nodes, depicting how the different nodes interact with each other over the network.

Deployment diagrams are useful for understanding the physical aspects of the system's architecture, showing how software components are distributed across hardware nodes. They are especially important in complex systems where various components need

to be distributed across multiple machines or devices for scalability, performance, and fault tolerance reasons.

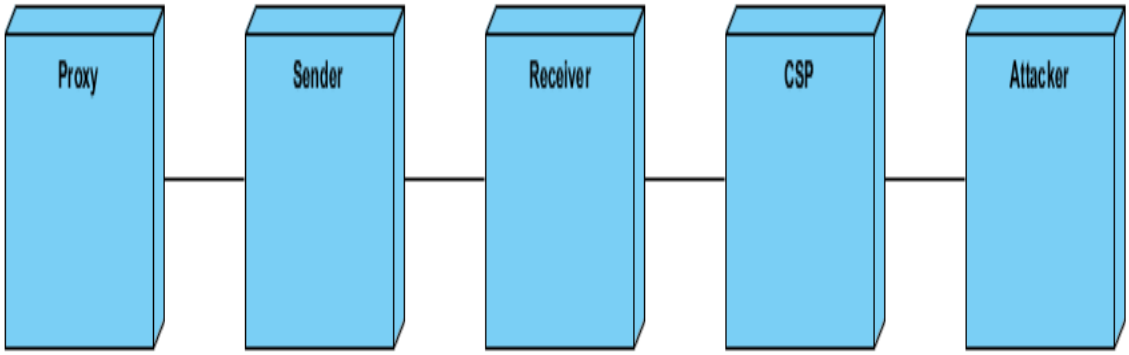


Figure 8: Deployment diagram

VII. RESULT

1. Proxy

Client Registration: Proxy needs to login with substantial qualifications into the framework. After effective login proxy is capable to enlist shipper and beneficiary data and moves login subtleties to collector/source through mail.

View Requests: Proxy can see the solicitation for records from the recipients and sends the secret entrance which unscramble the encoded data utilizing mail.

Logout: At last, logout from the framework.

Fig 9: - Proxy

2. Sender

Login: Sender will login with credentials which are sent by the proxy.

Upload Files: Just sender upload files after uploading he /she performs file encryption, trapdoor on file to secure it.

View Files: View files which are transferred by him.

Delete Files: At long last delete files which are not expected to him.

Logout: Logout from the framework.

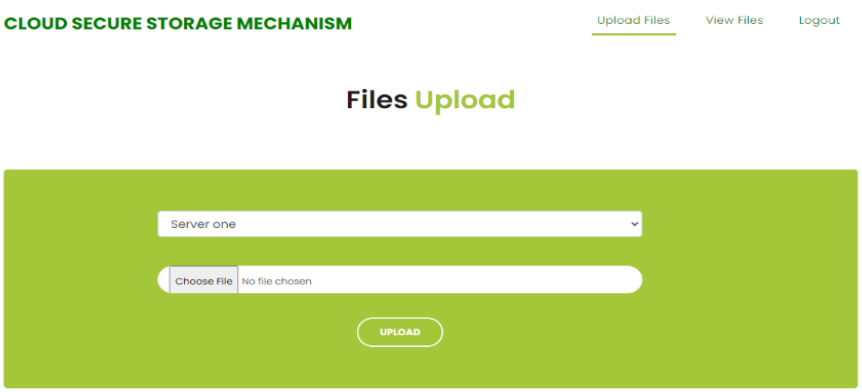


Figure 10: Upload File

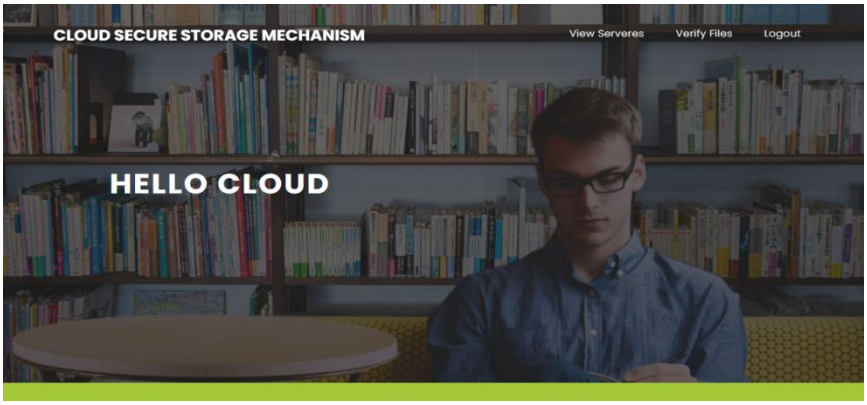
3. Cloud Service Provider

Login: Login with default values.

View Files: Cloud service provider have access to View all Files from all servers.

Verify Files: Verifies files which are attacked and secures files in servers.

Logout: Logout from the site



VIII. CONCLUSION

The paper presents a solution to address the issue of cloud information spillage coming about because of both administration carelessness and malignant assaults at the capacity layer. The proposed arrangement, CSSM (Cloud Secure Capacity Component), consolidates information dispersal and encryption advances to improve information security and forestall unapproved admittance to client information.

By distributing data fragments across different storage locations and encrypting them, CSSM effectively protects sensitive information from being compromised by hackers or other malicious entities. The experimental results demonstrate that CSSM successfully prevents client dataspillage at the disseminated storage layer . Additionally, the increased time overhead caused by the security measures is deemed acceptable by users in terms of performance, ensuring a balance between security and efficiency.

The paper presents a practical approach to address storage security concerns, focusing on preventing data leakage from cloud storage. CSSM not only protects user data effectively but also ensures the secure management of cryptographic materials.

Overall, the proposed CSSM provides a viable solution to improve the security of cloud data storage and prevent potential data breaches caused by unauthorized access or malicious activities, ensuring data confidentiality and integrity in a distributed storage environment.

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] 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

