SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V11I6P108

Volume 11 Issue 6, 53-60, June 2024

Original Article

A Model for Data Confidentiality and Integrity during Replication in Fog & Edge Computing Environment

Alalibo, H.[1], Bennett, E.O[2]

Department of Computer Science, Rivers State University, Port Harcourt, Nigeria.

[2]*Corresponding Author : bennett.okoni@ust.edu.ng*

Received: 24 April 2024 Revised: 29 May 2024 Accepted: 12 June 2024 Published: 30 June 2024

Abstract *- In the study of confidentiality and integrity during replication in fog and edge computing environments, the SHA-256 hashing algorithm and the PyCryptodome library played crucial roles. The SHA-256 algorithm, a member of the SHA-2 family, was employed for its robust cryptographic hash function, ensuring data integrity by generating a unique, fixed-size 256-bit hash value for every input. This characteristic was pivotal in detecting any alterations to the data, thereby preserving its integrity during transmission and storage. The PyCryptodome library, an extensive suite of cryptographic functions, was utilized to implement the SHA-256 algorithm effectively. This library provided the necessary tools to integrate advanced cryptographic techniques into the system, enhancing the security of data replication processes. By leveraging these technologies, the study aimed to establish a secure framework that safeguarded data confidentiality and integrity across distributed fog and edge computing environments. The results demonstrated the feasibility and efficacy of using SHA-256 and PyCryptodome in mitigating potential security threats, thereby contributing to the development of more resilient and trustworthy computing infrastructures. Object-Oriented Development (OOD) is employed to decrease development schedules, reduce resource demands, and improve code reusability. Python serves as the programming language, while MySQL functions as the database.*

Keywords *- Ensuring data confidentiality, Integrity assurance, Secure data replication, Encryption techniques, Access control mechanisms, Data integrity verification, Privacy-preserving strategies.*

1. Introduction

In the dynamic fog and edge computing landscape, protecting data privacy and authenticity while replicating it has become increasingly important. A growing number of Internet of Things (IoT) devices and edge computing nodes are driving up the amount of data created and processed at the network's perimeter. Data intrusions are a major concern in today's data-driven digital economy. Unfortunately, data breaches of various severity have affected many firms. As a result of the intrusions, third parties have gained malicious access to critical customer information, putting those people in danger. Data security vulnerabilities pose a significant threat to Nigeria due to the country's large population of over 200 million people. The country has seen a proliferation of cyber events, which have affected both public and private sector organizations. Data invasions in Nigeria saw a significant uptick in the first quarter of 2023, according to cybersecurity firm Surfshark. An astounding 82,000 have occurred, a 64% increase over the previous total. While traditional cloud computing stores and processes data in a central location, fog and edge computing move these tasks closer to the data's original point of origin. This dispersed nature offers benefits such as lower latency and bandwidth utilization. However, this dispersed nature also presents significant challenges in terms of data security. The suggested method employs an edge-and-fog computing environment to preserve data integrity and secrecy throughout replication. Information confidentiality hinges on granting access to only authorized workers.

2. Related Works

[1] An all-encompassing plan to address edge computing data integrity issues was put up by Li et al. (2021). Aside from a thorough experimental assessment, they also performed a systematic literature analysis and suggested a new integrity auditing procedure that uses cryptographic methods like Merkle hash trees. To highlight improvements in efficiency and security, the essay compared the suggested alternative to current techniques. However, issues with scalability, dynamic data updates, real-world implementations, user privacy, and interaction with modern technology have rendered the inquiry unfinished. The solutions that were suggested were tested in some situations, but how well they could handle larger, more complicated systems with different kinds of data and devices was not addressed. Furthermore, the study might benefit from more thorough real-world case studies to validate the solutions' viability and effectiveness in different deployment settings. Future studies should explore the integration of suggested auditing systems with emerging technologies such as blockchain and artificial intelligence to enhance security and efficiency further. In order to improve data integrity verification using aggregate signatures, Li et al. (2021) applied an approach. In order to ensure security

qualities, we first performed a literature analysis to identify the gaps and limitations in existing approaches. Then, we used aggregate signatures to build a unique integrity auditing protocol, which we mathematically formulated. Thorough simulations were carried out to evaluate the performance of the proposed strategy, with a major focus on its computational and communication efficacy. However, the inquiry revealed numerous gaps in existing knowledge. These include things like handling dynamic data, considering user privacy, being able to scale to large-scale systems, and being practical for implementation in the real world. Additional study is needed to identify effective mechanisms for managing dynamic data changes since the technique mainly focuses on static data integrity. To fully understand the practical difficulties and effectiveness in different operating settings, the study is missing real-world deployment and validation. To further enhance operational efficiency and security, future research might explore the possibility of integrating the proposed aggregate signature system with emerging technologies like AI and blockchain.

[2] Data security and privacy in cloud storage settings may be enhanced with the help of the systematic methodology suggested by Ming et al. (2019). A certificate-less Policy Decision Point (PDP) protocol, zero-knowledge protocols, randomization methods, and certificate-less encryption are all part of the methodology. The approach sought to fix major escrow problems with legacy systems and do away with the requirement for certificates. To further ensure that no data is leaked during verification, it uses zero-knowledge proofs. A comprehensive security analysis and performance review were conducted on the suggested approach to confirm its claims and measure its effectiveness. Nevertheless, there are still supplementary domains that necessitate investigation, such as the ability to manage massive environments efficiently, deal with dynamic data operations, put the technology to use in practical situations, and guarantee user privacy beyond integrity checks. We need more study to understand the practical issues and performance in real-world operational settings and to create and appraise strategies for managing dynamic data operations. Future studies should look at ways to combine the certificate-less PDP system with new technology, including blockchain and artificial intelligence, to make it more secure and efficient.

[3] Data replication and consistency in fog computing settings can be enhanced by the methods presented by Naas et al. (2021). The methodology includes data insertion strategies, measures for performance, methods for managing consistency, and assessment based on simulations. Using the iFogSim simulator, the authors put their plans into action and evaluated them. They focused on integrating ways to reduce service latency and assure data integrity across geo-distributed fog nodes. However, there are certain gaps in our understanding of the approach that affect its applicability and use. These include issues with energy consumption, scalability for different Internet of Things (IoT) scenarios, dynamic data management, and the incorporation of security measures. Additional research is needed to confirm these tactics in varied and large-scale IoT installations; however, the study did show gains in latency reduction. Without thoroughly covering security elements like data encryption and access control during replication procedures, the technique mainly focuses on consistency and replication. Furthermore, dynamic data operations are not well investigated.

[4] Dastjerdi et al. (2016) explore fog computing in depth. The authors highlight its concepts, architectures, and many applications. The authors reviewed the literature to lay the groundwork for fog computing and its guiding principles, with an emphasis on how it emerged as a response to cloud computing's shortcomings in latency-sensitive applications like the Internet of Things. They presented a comprehensive reference architecture for fog computing, detailing all its components and their interrelationships. In order to demonstrate how fog computing might alleviate network congestion and latency, the article looked at several use cases, such as health monitoring, emergency response, and smart city applications. The article compared fog computing with regular cloud computing based on performance parameters. The focus was on improving the quality of service for real-time applications, optimizing bandwidth, and reducing latency. However, the study acknowledged certain gaps in our understanding, including issues with scale and practical application, standards and interoperability, privacy and security, energy efficiency, and dynamic resource management. By filling these gaps, fog computing can become a more feasible and effective alternative for applications that are sensitive to latency and resource consumption.

[5] In their 2023 publication, Daoud et al. laid forth a thorough approach to protecting fog computing systems through efficient use of resources. The processes of monitoring, risk assessment, resource management, and access control are essential. The technique used a risk model to determine users' reliability, keep track of their actions, distribute resources effectively, and implement eXtensible Access Control Markup Language access controls. However, the inquiry revealed several knowledge gaps. These included issues with scalability, energy efficiency, interoperability, and dynamic adaptability, as well as the need for practical application. Only testing and implementation in the real world can determine the effectiveness and difficulties in various situations. Energy consumption optimization, cloud and fog system integration, and adaptive methods for real-time resource allocation and danger mitigation should be the focus of future studies. By addressing these shortcomings, we can enhance the robustness and applicability of security frameworks in fog computing environments, thereby protecting more applications and managing resources more efficiently.

[6] Ometov et al. (2022) compared and analyzed security concerns in the context of cloud, edge, and fog computing paradigms using a comprehensive literature

review. The process involved curating relevant literature from many sources. The writers subsequently classified it based on the computer paradigm and certain security factors, including privacy, availability, confidentiality, and integrity. The writers compared the security procedures, risks, and mitigation strategies of cloud, edge, and fog computing. They highlighted the main threats and weaknesses specific to each paradigm, as well as the ways in which existing solutions address these issues. The study identified emerging dangers and sectors with inadequate current solutions as research gaps that required filling while also uncovering current trends in security research. The survey also revealed numerous unresolved issues, such as managing heterogeneity, ensuring the effectiveness of security solutions in real-world scenarios, enhancing the interoperability of security protocols, adapting dynamically to threats, and safeguarding user privacy. Future studies should fill these gaps. By filling these gaps, future studies can safeguard cloud, edge, and fog computing systems against emerging threats and boost their security and dependability, making them more capable of handling the growing needs of various applications.

[7] Fatimeh et al. (2024) examined data replication strategies in cloud, fog, and edge computing through a thorough literature review. This research compiles relevant studies from scholarly publications, sorts them according to computing paradigm and data replication technique, and then compares them according to consistency, availability, and latency. By employing this method, we can discern patterns in data replication studies and pinpoint areas that require attention. The presentation of significant insights revealed knowledge gaps in several areas, including real-world application, energy efficiency, interoperability, security and privacy, scalability and performance, and dynamic adaptation. More study of real-world applications, especially diverse ones, is needed to explain performance and practical issues. Research on scalability in big fog and edge networks is still in its infancy, and energy efficiency is a must for sustainable edge computing. Further study is necessary to address the challenging issue of interoperability and standardize data synchronization and replication processes. Privacy, security, and the capacity to adjust to changing network circumstances and user needs must be given top priority. Future studies may address these shortcomings to make data replication in cloud, fog, and edge computing more secure, sustainable, and effective.

[8] Torabi et al. (2020) use a systematic review methodology to examine and classify data replication strategies in fog computing scenarios. There is a lack of information on several fronts, including interoperability, privacy and security, scalability, energy efficiency, and practical application data. The study highlights research on safe and privacy-preserving replication algorithms, scalable solutions for large-scale deployments, and thorough studies of the trade-offs between data replication and energy use. It further highlights the significance of flexible algorithms that can swiftly react to changes in user behavior and network circumstances. The research concludes that closing the gaps in fog computing can lead to more secure results and more dependable and effective data replication procedures, thereby improving the overall functionality and reliability of fog-based systems.

[9] Kaliyaperumal et al. (2023) came up with a new way to make sure data stays intact in fog computing settings. The approach included breaking down large amounts of IoT data into smaller, more manageable chunks and then using fuzzy clustering to find elements of the data that were comparable. Data integrity verification is the backbone of the technique; it improves efficiency and reduces costs by removing the need for extra client-side information storage. However, the study may benefit from more thorough performance measurements and consideration of scalability under different scenarios. The technique does not address additional security considerations like user authentication and data confidentiality. Furthermore, there are limited studies that compare this technique to other data integrity verification methods. Only real-world implementation and testing can confirm the study's applicability and usefulness in live fog computing systems, as it primarily relied on theory and simulations. The study did not address potential consequences or trade-offs related to user-side computing resources and network bandwidth. Dynamic data management requires more research.

[10] Research by Tian et al. (2019) examined a public auditing approach that uses privacy-preserving tactics and efficient verification processes. They assessed the protocol using security analysis and performance measurements. In addition to improving data storage security, the protocol fills a need for resolving latency difficulties. Future research should prioritize optimizing auditing protocols for fog computing settings with low latency. To allow peripheral computing systems and fog to change dynamically, adaptive security frameworks are required. The smooth functioning of cloud, fog, and edge paradigms depends on unified security standards. Scalability optimization of models is essential, as are real-time methods for ensuring data integrity. Future research should primarily focus on finding ways to speed up audits without increasing delays.

3. Methodology

Object-Oriented Analysis and Design Methodology (OOADM) was used to undertake the implementation of the suggested system design. The Object-Oriented Systems Analysis and Design Methodology was a system approach to information system analysis and design that focused on software objects and how they interacted with the system and its surrounding environment. The study's goals were to abstract components, give them names, and then organize their activities into class abstractions. The next step in developing these class abstractions was to determine their function within the system and the individuals responsible for carrying them out.

3.1. Analysis of the System

Once a user has created an account, the system will immediately produce a private key and store it in the Edge

node. Upon document submission, the system will automatically create a key for file signatures. To ensure data integrity and prevent data replication, the user will first download the file and then check the document's digital file signature value. The use of distributed databases, fog, and edge computing ensured data redundancy and accessibility across multiple nodes. Data encryption was used to transform plaintext data into ciphertext, which ensures that it cannot be deciphered by unauthorized individuals. All data fragments that were duplicated across the network were encrypted before transmission. The target node decrypted the data back to its original state upon arrival. This approach would ensure that the data remained inaccessible and protected from malevolent actors, even if intercepted during replication. A cryptographic hash function digitally signed each data transport before replication. The data had remained unchanged during transmission, as verified by this signature. The destination node confirmed the data's integrity and authenticity upon receiving the digital signature. Every node has two keys: one for data signing and one for verifying signatures. This architecture allowed for the secure signing and validation of data, preventing unwanted adjustments during replication. To ensure data security and integrity during replication, the SHA-256 hashing method was used. To implement SHA-256 hashing, we used the PyCryptodome module, which is a self-contained Python package containing low-level cryptographic primitives. See Figure 1 for an illustration of the system's architecture.

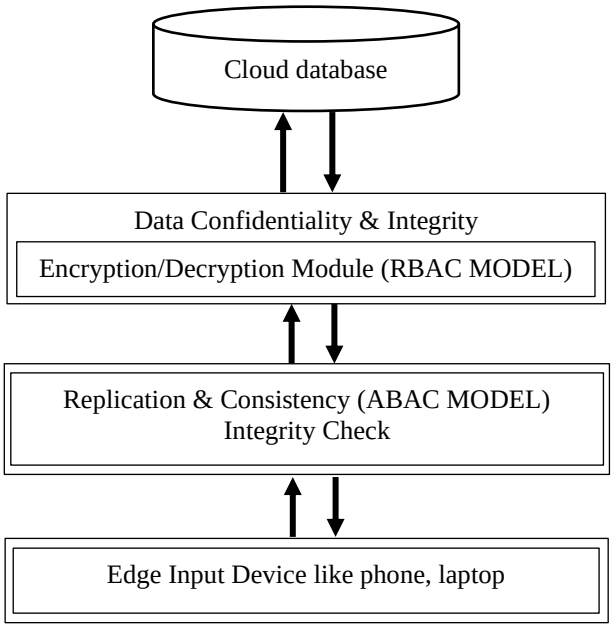

Fig. 1 System architecture

3.2. Mathematical Model

An RSA public-key / private-key pair can be generated by the following steps:

The file is uploaded into the cloud database after being divided into small pieces denoted as F. label equation is:

$$F_i=(f_1,f_2,f_3,f_n)^n \quad (1)$$

The database requires the generation of two large, random primes, e and d, for file encryption. The encryption operation in the RSA crypto-system is exponential to the eth power modulo n:

$$c=ENCRYPT(m)=eMODd \quad (2)$$

The message m is usually a key that is properly structured and meant to be shared. A conventional technique employs the shared key to encrypt the actual message. This method can encrypt a message of any length with just a single exponentiation. Amplification of the dth power modulo n is used to carry out the decryption operation.

$$m=DECRYPT(m)=e^dMODd \quad (3)$$

Encryption and decryption are inherently inverse operations due to the connection between the exponents e and d. Therefore, the decryption procedure recovers the initial message m. Applying the decryption operation to a message—which entails raising it to the dth power—is the process of digitally signing it.

$$s=SIGN(m)=e^dMOD \quad (4)$$

Applying the encryption procedure on the digital signature and comparing the output with the message or recovering it can verify it.

$$m=VERIFY(s)=s^eMODd \quad (5)$$

class RBAC: roles = {}

permissions = {}

3.3. Algorithm of the System

def assign_permission(self, role, permission): # Assign permission to a role roles[role].add_permission(permission)
class ABAC: attributes = {} policies = [] def assign_attribute(self, user, attribute): # Assign attribute to a user attributes[user].add(attribute)
def add_policy(self, conditions, actions):
 # Add a policy with specified conditions and actions policies.append({"conditions": conditions, "actions": actions}) class PEP
def enforce_policy(self, user_attributes, requested_action):

 # Enforce policies based on user attributes and requested action decision = PDP.evaluate_policies(user_attributes, requested_action) if decision == "Allow":
 # Grant access
return "Access Granted" else:
 # Deny access
return "Access Denied"
class PDP: policies = []
def add_policy(self, conditions, actions):
 # Add a policy with specified conditions and actions policies.append({"conditions": conditions, "actions": actions})
def evaluate_policies(self, user_attributes, requested_action):

 # Evaluate policies based on user attributes and requested action for policy in policies:
if satisfies_conditions(user_attributes, policy["conditions"]) and \ requested_action in policy["actions"]:
return "Allow" return "Deny"
def satisfies_conditions(user_attributes, conditions): # Check if user attributes satisfy the conditions for condition in conditions:
if condition not in user_attributes: return False
return True
class DataReplication:
def replicate_data(self, data, destination):
 # Replicate data to the specified destination
 # Include mechanisms for ensuring data confidentiality and integrity encrypted_data = encrypt(data)
hash_value = calculate_hash(data)
 # Send encrypted_data and hash_value to the destination # Verify integrity at the destination
def encrypt(data):
 # Implement encryption algorithm pass
def calculate_hash(data):
 # Implement hash calculation algorithm pass
 # Static Analysis
 # - Perform consistency checks on RBAC and ABAC configurations # - Analyse policies for completeness and conflicts
 # Dynamic Analysis
 # - Simulate access requests with varying user attributes and actions # - Evaluate the dynamic behaviour of policies in PDP

3.4. Sequence Diagram

The system will save the user's data, including their IP address. Users are prompted to choose a document type, such as PDF or DOC, and the system will automatically utilize RSA to encrypt the file. It is necessary for the user to provide the private key while decrypting, and the server will then check if it matches the current key.

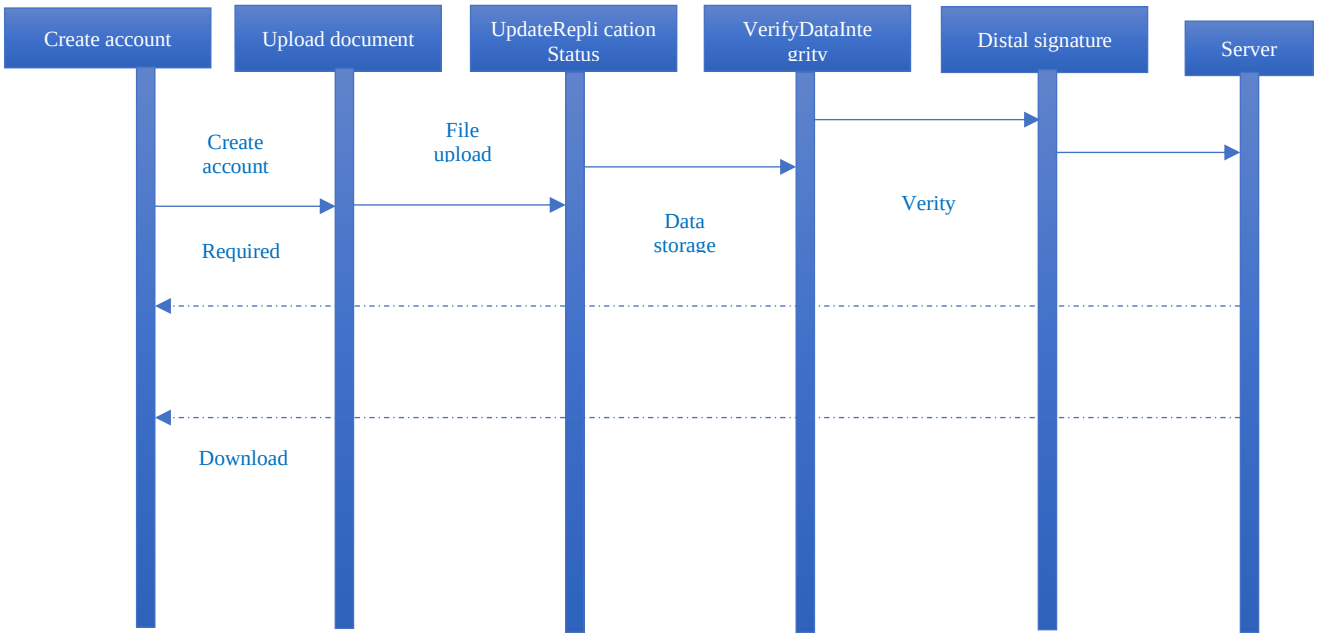

Fig. 2 Sequence diagram

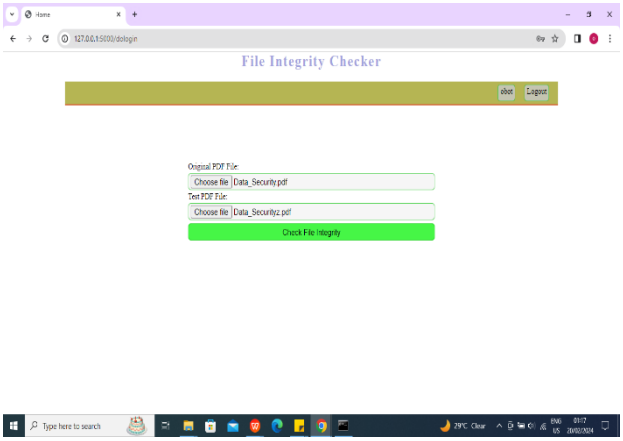

Fig. 3 File upload page

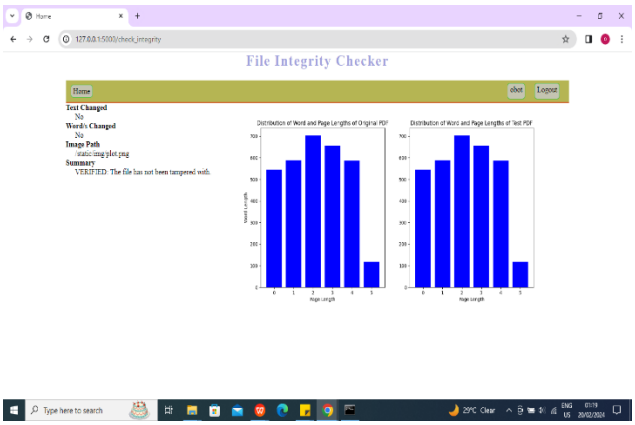

Fig. 4 Original file and download file comprise

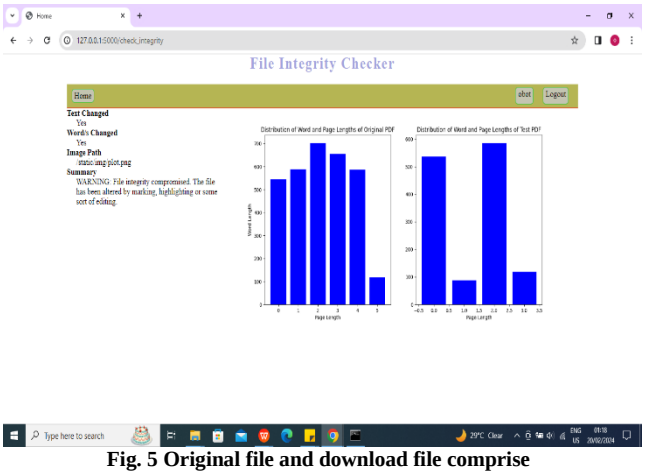

Fig. 5 Original file and download file comprise

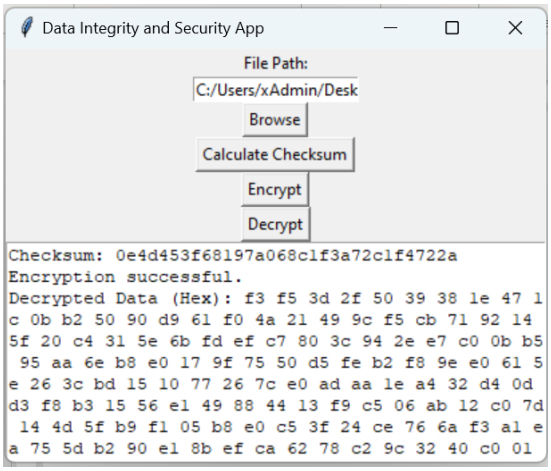

Fig. 6 File encryption page

Table 1. Encryption file result

File (MB)	Size Checksum (s)	Integrity (s)	Hash (s)	Encryption (s)	Decryption (s)	Signature (s)
10	0.021	0.032	0.015	0	0	0
100	0.238	0.235	0.248	0.016	0	0.018
60	0.145	0.147	0.146	0.016	0	0.014
10	0.016	0.011	0.018	0	0	0.016
30	0.091	0.08	0.07	0.017	0	0
40	0.094	0.094	0.086	0	0	0.017
110	0.303	0.267	0.256	0.033	0	0.014
90	0.236	0.239	0.22	0.022	0.005	0.036
70	0.155	0.161	0.16	0.001	0.015	0.012
40	0.102	0.095	0.1	0	0	0.018
80	0.171	0.187	0.184	0.031	0.002	0.014
120	0.282	0.285	0.285	0.022	0.008	0.035

4. Discussion of Result

Figures 3, 4, and 5 display the output results of the system. For the distributed systems to work, we had to replicate data without changing it. It was crucial to ensure precise data replication in multiple locations before uploading a file to the fog or edge nodes. The process began with creating a unique hash value for the source file. This hash value functioned as a digital identity, preserving the precise content of the file at the time of creation. It would be simple to spot changes or inconsistencies by comparing the downloaded file's hash value to the original. During replication, some edge and fog nodes received the original file, which they saved after downloading. Each node recalculated the hash value of the downloaded file as part of the integrity check. We compared the hash values to ensure the file's integrity and prevent any modifications. However, the disparity indicated a problem during the file's transmission. Several steps were taken to address any integrity issues. Using secure transmission protocols like HTTPS and SSL/TLS reduced the possibility of interception and interference by encrypting data in transit.

Furthermore, we implemented redundancy measures, such as storing the file in multiple locations, to ensure the recovery of the original content in the event of a compromised location. We also instituted periodic integrity checks further to guarantee consistent monitoring of the files' state. We compared the original hash values of the cached files with their current hash values to detect any unexpected changes. In the event that inconsistencies were found, administrators were notified automatically so that they could fix the file and restore its integrity.

Figure 6 and Table 1 showed the encryption technique for data confidentiality during replication. We used the SHA-256 hashing method to safeguard sensitive information from unwanted access and intrusions during replication. Implementing SHA-256 encoding for data encryption included several important processes. Firstly, we applied a thorough encryption method to the data that we were replicating. We selected the cryptographic hash function SHA-256 due to its strong security characteristics and resistance to collision attacks. It is extremely unlikely that unauthorized parties will be able to decipher the original material from the hash because this technique transformed it into a fixed-size, 256-bit hash value. Several nodes in the fog and edge computing networks relayed the encrypted data during replication. By comparing the hash

values, any node with the necessary decryption keys can verify that the data received is genuine and intact. This ensured the privacy and authenticity of the copied data by immediately revealing any alterations or tampering during transit. In addition, SHA-256 encoding prevented attacks that may have targeted the replication process, which was an extra safeguard. After transforming the data into a hash value, it became resistant to plaintext assaults. This transformation prevented malicious actors from intercepting and deciphering unencrypted information. This method ensured the safety of sensitive data in a distributed computing setting by drastically reducing the possibility of data invasions. Deploying SHA-256 hashing for data encryption in the fog and edge computing environments successfully demonstrated a complete solution to preserving secrecy throughout replication. By harnessing the power of cryptographic hash functions, the system successfully secured sensitive data, ensuring its integrity and security throughout the replication process. This implementation highlighted the importance of using advanced encryption methods to tackle the specific problems caused by the dispersed nature of fog and edge computing networks.

4.1. Evaluation of Result

The comparison of the SHA-256 hashing algorithm and the PyCryptodome library against other techniques for ensuring confidentiality and integrity during replication in a fog and edge computing environment highlighted several key aspects.

Cryptographic applications chose SHA-256, a member of the SHA-2 family, for its robustness and widespread acceptance. Its strength lay in its fixed output length of 256 bits, which provided a high level of security against collision and preimage attacks. This made it an ideal candidate for maintaining data integrity, as any alteration in the data resulted in a significantly different hash, making unauthorized modifications easily detectable. Additionally, the deterministic nature of SHA-256 ensured that the same input always produced the same hash output, facilitating consistent data replication across multiple nodes in the fog and edge environments.

The PyCryptodome library was used to enhance the implementation of SHA-256 further. PyCryptodome, a self-contained Python library, offered a comprehensive suite of cryptographic primitives and provided a user-friendly interface for implementing SHA-256. Its compatibility with Python's standard libraries, as well as the absence of external dependencies, streamlined the development process. PyCryptodome's optimized performance and regular updates incorporated the latest security features and bug fixes, ensuring a reliable and secure framework for hashing operations.

In contrast to other techniques, SHA-256, combined with PyCryptodome, outperformed alternative hashing algorithms like MD5 and SHA-1, which had known vulnerabilities such as susceptibility to collision attacks. The choice of SHA-256 addressed these weaknesses, offering a more secure alternative. Furthermore, when compared to newer algorithms like SHA-3, SHA-256's established track record and extensive documentation made it a more practical choice for immediate implementation, especially in environments where time and resource constraints were significant factors.

The evaluation also considered symmetric encryption methods and their role in maintaining confidentiality. While algorithms like AES provided robust encryption, the focus on hashing with SHA-256 was particularly relevant for ensuring data integrity during replication. When compared to the heavy computational load of encryption and decryption, hashing operations are much lighter. This made SHA-256 a better choice for fog and edge devices with limited resources.

After a thorough analysis, it was found that using SHA-256 along with the PyCryptodome library was the best way to protect privacy and integrity while copying data in fog and edge computing settings. This approach effectively balances security, performance, and ease of implementation, making it a preferred choice over other cryptographic techniques.

5. Conclusion

All things considered, the PyCryptodome library's use of the SHA-256 hashing technique to ensure privacy and authenticity during replication in cloud and edge computing settings worked like a charm. The strong cryptographic hash functions of the SHA-256 algorithm protected the data from prying eyes and accidental changes. We accelerated the integration process by utilizing the PyCryptodome module's features, which enabled safe and efficient handling of data replication across dispersed nodes. By combining SHA-256 with PyCryptodome, we were able to solve the fundamental issues with privacy and security in the fog and edge computing models. SHA-256's cryptographic strength ensured data integrity, and its sensitivity to changes allowed for easy detection of any alterations. PyCryptoDome also made it easier to deploy these security features, which improved the system's overall dependability. This strategy not only secured data during transmission but also boosted trust in the distributed computing system, reducing the possibility of security breaches. Modern computer environments necessitate strong cryptographic methods, according to the study. PyCryptodome and other comprehensive libraries are essential for achieving high data security standards. Finally, this study demonstrated the practicality and effectiveness of combining PyCryptodome with SHA-256 to protect data privacy and integrity in cloud and edge computing environments. The writers state categorically that they have no competing interests.

References

[1] Bo Li et al., "Inspecting Edge Data Integrity with Aggregate Signature in Distributed Edge Computing Environment," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2691-2703, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Yang Ming, and Wenchang Shi, "Efficient Privacy-Preserving Certificateless Provable Data Possession Scheme for Cloud Storage," *IEEE Access*, vol. 7, pp. 122091-122105, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Mohammed Islam Naas et al., "IoT Data Replication and Consistency Management in Fog Computing," *Journal of Grid Computing*, vol. 19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] A.V. Dastjerdi et al., "Chapter 4 - Fog Computing: Principles, Architectures, and Applications," *Internet of Things*, pp. 61-75, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[5] Wided Ben Daoud et al., "Fog Computing Network Security Based on Resources Management," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Esmaeil Torabi, Mostafa Ghobaei-Arani, and Ali Shahidinejad, "Data Replica Placement Approaches in Fog Computing: A Review," *Cluster Computing*, vol. 25, no.5, pp. 3561-3589, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[7] Uma Maheswari Kaliyaperumal, Mary Saira Bhanu Somasundaram, and Nickolas Savarimuthu, "Partitioning-Based Data Sharing Approach for Data Integrity Verification in Distributed Fog Computing," *International Journal of Engineering and Technology Innovation*, vol. 13, no. 2, pp. 160–174, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Hui Tian et al., "Privacy-Preserving Public Auditing for Secure Data Storage in Fog-to-Cloud Computing," *Journal of Network and Computer Applications*, vol. 127, pp. 59-69, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] A. Fatimeh, B. Ali, and C. Smith, "Analysis of Cloud, Fog, and Edge Computing Data Replication Methods: A Systematic Literature Review," *Journal of Computing Research*, vol. 29, no. 2, pp. 145-167, 2024.

[10] Aleksandr Ometov et al., "A Systematic Literature Review on Security Challenges in Cloud, Edge, and Fog Computing," *Journal of Cloud Computing*, vol. 11, no. 3, pp. 275-302, 2022.

[11] Fatemeh Karamimirazizi, Seyed Mahdi Jameii, and Amir Masoud Rahmani, "Data Replication Methods in Cloud, Fog, and Edge Computing: A Systematic Literature Review," *Wireless Personal Communications*, vol. 135, pp. 531-561, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Aleksandr Ometov et al., "A Survey of Security in Cloud, Edge, and Fog Computing," *Sensors*, vol.22, no.3, pp. 1-27, 2022. [CrossRef] [Google Scholar] [Publisher Link]