International Journal of Multidisciplinary Innovative Research
ISSN: 2583-0228 Volume 6, Number 1 (Jan' 2026) pp. 28-37
© CIIR, Noida, INDIA (www.ciir.in)
https://www.ijmir.org
Article Received: 10 Oct' 2025
Final Revision: 08 Nov' 2025
Accepted for Publication: 24 Nov' 2025

Research Article

OPEN ACCESS

Enhancing Cloud Data Security Through AES Encryption and SHA-256 Integrity Verification

Mayank Joshi, Anuja Chincholkar, Anuj Londhe, Parth Gole and Saniya Mirgale

Department of Computer Science & Engineering, MIT ADT University, Pune, India.

Email: anuja.chincholkar@mituniversity.edu.in, anujlondhe01@gmail.com, mayankjoshi2303005@gmail.com, goleparth824@gmail.com, saniyamirgale@gmail.com*

ABSTRACT: Cloud computing has emerged as a transformative technology that provides scalable, flexible, and cost-effective solutions for data storage, processing, and management. Despite its extensive adoption, it continues to face significant security challenges concerning the confidentiality, integrity, and availability of data. Threats such as data breaches, unauthorized access, and malicious tampering are particularly critical in sensitive domains like healthcare, finance, and e-learning, where data privacy is paramount. To mitigate these issues, this research introduces a hybrid cryptographic framework that combines the Advanced Encryption Standard (AES-256) for secure data encryption with the Secure Hash Algorithm (SHA-256) for integrity verification. The proposed system was implemented and tested on Amazon Web Services (AWS), leveraging S3 for data storage and Identity and Access Management (IAM) for access control. Various file types—including text, image, and multimedia—were encrypted on a MacBook Air M2 prior to uploading. Performance metrics such as encryption and decryption time, hashing duration, upload/download latency, and system resource utilization were measured, along with security evaluations involving hash verification, unauthorized access detection, and tampering simulations. Experimental results demonstrate that the integration of AES-256 and SHA-256 provides a robust, scalable, and efficient data protection mechanism for cloud environments. This dual-layer framework effectively ensures both confidentiality and integrity without imposing substantial performance overhead, making it suitable for real-world deployment. Furthermore, the study addresses existing research gaps in dynamic data protection and large-scale data optimization while paving the way for future advancements such as multi-cloud integration, adaptive access control, and intelligent threat detection mechanisms.

KEYWORDS: Cloud Security, Data Security, AES-256, SHA-256, Cryptography, Data Integrity, Data Confidentiality, Hybrid Encryption, Cloud Storage, Amazon Web Services (AWS), Identity and Access Management (IAM), Hash Verification, Data Protection.

INTRODUCTION

Cloud computing has become one of the most important technologies in today's digital world. It allows individuals and organizations to store, process, and manage large amounts of data on remote servers instead of relying on local hardware. Services like Amazon Web Services (AWS), Google Cloud, and Microsoft Azure provide easy access to data from anywhere, at any time, and allow organizations to save costs on maintaining their own physical infrastructure. Cloud computing also supports collaboration, scalability, and faster deployment of applications. However, while cloud services offer many benefits, they also introduce serious security challenges, especially when it comes to storing sensitive or personal data. Data stored in the cloud can be vulnerable to unauthorized access, data theft, tampering, accidental deletion, and insider threats. These risks make it essential to implement strong security measures to protect cloud data and ensure that it remains safe, private, and reliable [1].

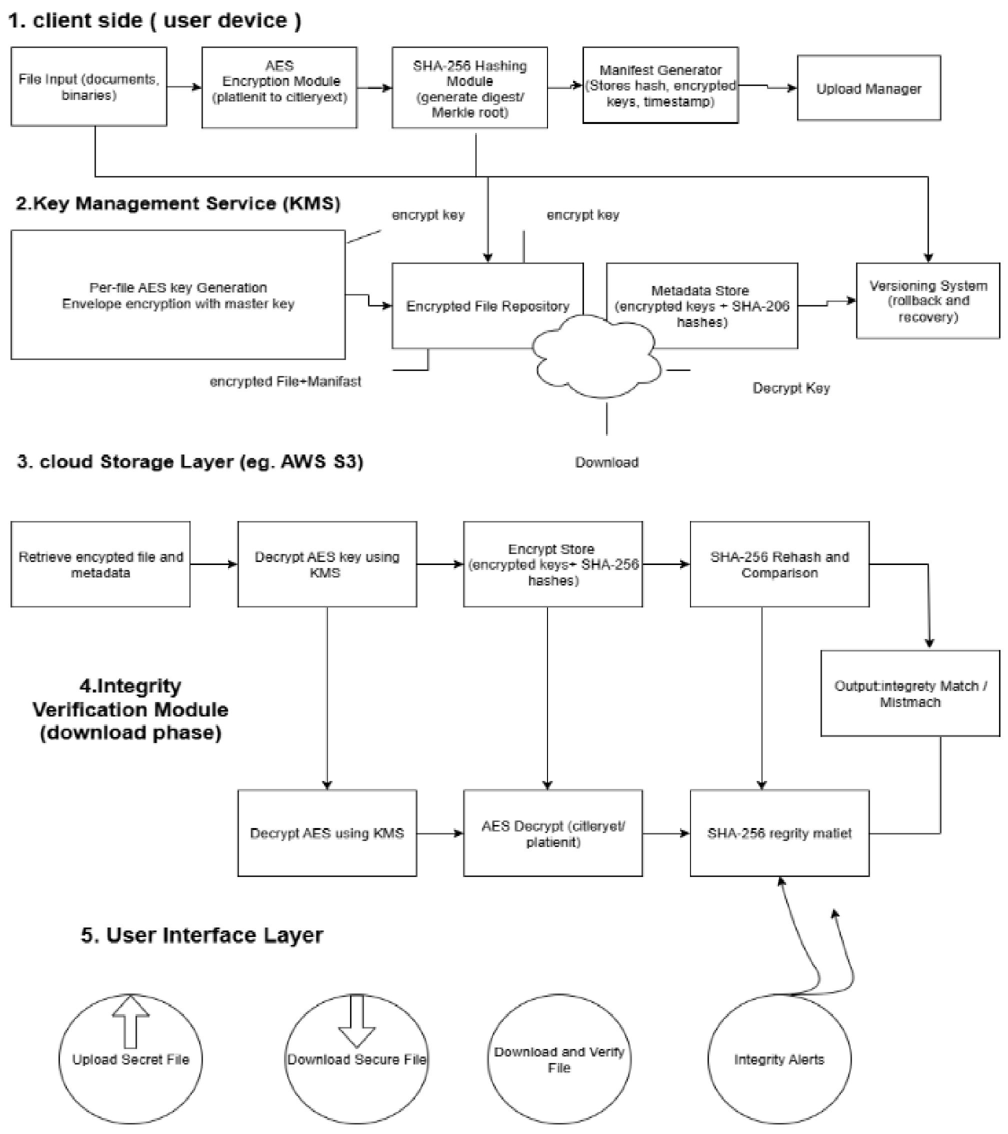

1. client side (user device)
File Input (documents, binaries)
AES Encryption Module (platienit to citleryext)
SHA-256 Hashing Module (generate digest/ Merkle root)
Manifest Generator (Stores hash, encrypted keys, timestamp)
Upload Manager
2.Key Management Service (KMS)
encrypt key
encrypt key
Per-file AES key Generation Envelope encryption with master key
Encrypted File Repository
Metadata Store (encrypted keys + SHA-206 hashes)
Versioning System (rollback and recovery)
encrypted File+Manifast
Decrypt Key
3. cloud Storage Layer (eg. AWS S3)
Download
Retrieve encyptled file and metadata
Decrypt AES key using KMS
Encrypt Store (encrypted keys+ SHA-256 hashes)
SHA-256 Rehash and Comparison
Output:integrety Match / Mistmach
4.Integrity Verification Module (download phase)
Decrypt AES using KMS
AES Decrypt (citleryet/ platienit)
SHA-256 regrity matiet
5. User Interface Layer
Upload Secret File
Download Secure File
Download and Verify File
Integrity Alerts

Figure 1: Proposed system architecture

One of the most effective ways to protect data in the cloud is through encryption. Encryption transforms readable data, called plaintext, into an unreadable format, called ciphertext, using mathematical algorithms and a secret key. Without the correct key, unauthorized users cannot read or access the data. Among different encryption techniques, the Advanced Encryption Standard (AES) is widely used because it is fast, secure, and efficient for handling large amounts of data. AES with a 256-bit key length, known as AES-256, provides very strong security, making it extremely difficult for attackers to break. While AES protects the data from being read by unauthorized users, it does not automatically ensure that the data has not been changed or tampered with during storage or transmission.

To ensure that data has not been altered, hashing techniques are used. A hashing algorithm takes data and produces a fixed-length unique value, called a hash or digest, which acts as a digital fingerprint of the data. Even a small change in the original data produces a completely different hash. The SHA-256 algorithm is a popular cryptographic hash function that is commonly used for verifying the integrity of files. By generating a SHA-256 hash of a file before uploading it to the cloud, users can later compare it with a hash of the downloaded file to detect any changes or tampering. This ensures that the data remains intact and trustworthy [2].

By combining AES encryption and SHA-256 hashing, a complete security framework can be created. In this system, files are encrypted on the client side before they are uploaded to the cloud. At the same time, SHA-256 hashes are calculated and stored as part of a manifest—a record of file information such as hashes, encryption metadata, timestamps, and cloud storage paths. When the file is downloaded from the cloud, it is decrypted locally, and its hash is compared with the stored hash to verify integrity. This approach ensures both confidentiality (protection from unauthorized access) and integrity (protection from tampering or corruption).

This project focuses on building a practical system that allows users to securely store and retrieve their files in cloud environments such as AWS S3. The system uses AES-GCM encryption for strong confidentiality and authenticated encryption, and SHA-256 for verifying data integrity. Key management is handled securely on the client side, and cloud features such as versioning are used to maintain backup copies and prevent data loss. The system is designed to handle files of different sizes efficiently, including large files, and to detect tampering through manifest checks.

LITERATURE SURVEY

Client-Side AES Encryption for Cloud Security

This emphasized the critical role of AES-based white-box cryptography in digital signature verification and data protection. Their research demonstrated that AES, as a symmetric encryption algorithm, is highly efficient for encrypting large volumes of data while maintaining strong resistance against cryptographic attacks. By using AES encryption, sensitive information can be securely transformed into ciphertext, making it unreadable to unauthorized users. This approach ensures confidentiality, which is essential for cloud storage where data is often accessible over the internet. The study also highlighted that proper implementation of AES can prevent common vulnerabilities such as brute-force attacks or unauthorized key disclosure. These findings support the methodology of this project, where AES encryption is applied on the client-side to secure files before uploading them to cloud storage [3].

Secure File Storage Using Hybrid Cryptography

It proposed a hybrid cryptographic model to secure cloud-stored data, combining encryption with integrity verification mechanisms. Their research showed that while AES is effective for fast and reliable encryption, additional techniques such as cryptographic hashing are necessary to ensure that files are not tampered with during storage or transmission. They emphasized that hybrid cryptography provides a balance between performance and security, allowing organizations to store sensitive data in the cloud without compromising accessibility or speed. This approach is directly relevant to this project, as it uses AES encryption for confidentiality and SHA-256 hashing for integrity verification, ensuring that users can detect any unauthorized modifications to their files [4].

Integrity Verification with Hashing

This focused on the application of SHA-256 hashing along with ECC (Elliptic Curve Cryptography) for secure and reliable file transfer. Their study demonstrated that hashing algorithms like SHA-256 are essential for maintaining data integrity because even a single-bit change in a file can be detected by comparing hash values. The research highlighted that cloud-stored data is vulnerable not only to external attacks but also to accidental modifications during storage or transmission. By calculating a SHA-256 digest of the data before upload and verifying it after download, users can ensure that the files remain unchanged. This methodology is incorporated into this project through manifest files that store SHA-256 hashes, allowing end-to-end integrity verification for all cloud-stored files [5].

Enhanced Cloud Storage and Authentication

It explored AES-based cloud authentication systems that combined secure storage with user verification and version control mechanisms. Their research showed that encryption alone is not enough; proper access management and versioning are also crucial to prevent unauthorized access and ensure recoverability of data in case of accidental deletion or corruption. This study emphasizes the importance of integrating encryption, access control, and cloud-specific features like AWS S3 versioning, which is implemented in this project. By using IAM policies and versioning, the project ensures that only authorized users can access encrypted files and that previous versions of files can be recovered if necessary [6].

Comprehensive Security Frameworks in Cloud Computing

This presented a comprehensive approach to securing sensitive cloud data, focusing on privacy preservation, encryption, and integrity assurance. Their work highlighted that secure cloud storage systems must combine multiple techniques, including client-side encryption, hashing for integrity, and careful key management, to provide reliable protection [7]. They also discussed the importance of maintaining logs and manifests to track file changes and ensure accountability. This aligns closely with the goals of this project, which implements AES encryption for confidentiality, SHA-256 for integrity, and manifests to track encryption metadata and hashes. Their research reinforces the need for a holistic approach to cloud data security that addresses multiple threats simultaneously, which this project adopts through an end-to-end encryption and integrity verification system [8].

Hybrid Cryptography and Authentication in Cloud Storage

The paper presented a secure cloud storage solution that not only focused on encryption but also emphasized better data accessibility and time efficiency. Their model highlighted the importance of balancing strong security with seamless user experience. Likewise, Abualkas and Bhaskari developed a hybrid ECC–AES approach with key management techniques to improve both security and performance [9]. These works reinforce the view that hybrid models, where AES is combined with other cryptographic or key management mechanisms, can provide stronger protection while ensuring usability in real-world cloud environments [10].

Sl.	Author(s)	Year	Key Contribution	Relevance to Project
1.	[8] R. H. Najm	2019	Proposed enhancing AES key generation using SHA-256 to improve cloud data security.	Shows the combined use of AES and SHA-256 to ensure confidentiality and integrity of cloud data.

2.	[9] V. L. Mehetre, P. Wakhare, S. Mukundrao Bhilegaonkar, Y. Deepak Sinkar	2024	Analyzed various cryptographic techniques, emphasizing AES encryption and SHA-256 hashing.	Supports project methodology for securing cloud files with AES and verifying integrity using SHA-256.
3.	[10] J.-F. Lai, S.-H. Heng	2022	Proposed secure file storage on cloud using hybrid cryptography and metadata tracking.	Demonstrates manifest-based approach for file integrity, relevant to SHA-256 hash storage in this project.
4.	[6] L. Megouache, S. Sadouni, O. Sadouni, M. Djoudi, A. Zitouni	2024	Discussed AES-based secure cloud authentication and storage with versioning and access control.	Supports use of AWS S3 versioning and IAM policies for secure cloud storage.
5.	[11] S. Akter, Md. A. R. Khan, F. Rahman, S. J. Soheli, N. J. Suha	2023	Proposed AES-RSA hybrid encryption for cloud security and key management.	Highlights importance of client-side encryption and key security, adapted in project using AES-GCM and derived keys.

PROPOSED METHODOLOGY

The methodology of this research focuses on developing a secure cloud data management system that uses advanced encryption techniques to protect sensitive information. The main goal is to ensure confidentiality, integrity, and secure access of data stored in the cloud. The approach combines practical experimentation, analysis, and simulation to design a reliable security framework [12]. The methodology of this research focuses on developing a highly secure and efficient system for storing and transmitting data in cloud environments using a hybrid cryptographic approach [13]. The process begins with the collection of digital data from users, which may include text files, images, videos, or other sensitive information that needs to be securely stored. Before uploading the data to the cloud, it undergoes a multi-layered encryption process to ensure maximum security [14]. Initially, the AES (Advanced Encryption Standard) algorithm is applied to encrypt the actual content of the files. AES is chosen due to its speed, reliability, and strong resistance to attacks, making it ideal for handling large volumes of data efficiently. Simultaneously, the AES encryption keys themselves are protected using the RSA (Rivest–Shamir–Adleman) algorithm [15], which employs asymmetric encryption to secure the keys and ensure that only authorized users with the correct private keys can access the data.

To maintain data integrity, the SHA-256 hashing algorithm is applied to generate a unique hash value for each file, acting as a digital signature to detect any unauthorized modifications [16]. After encryption and hashing, files and their encrypted keys are uploaded to cloud storage with strict access control managed through IAM. During retrieval, the RSA algorithm decrypts the AES keys, which are then used to decrypt the original files. To further strengthen data security and maintain integrity, each file is processed using the SHA-256 hashing algorithm, which generates a unique fixed-length hash for every file. This hash acts as a digital fingerprint that can be verified during retrieval to detect any unauthorized alterations or corruption of the data [17]. After encryption and hashing, the files, along with their securely encrypted AES keys, are uploaded to the cloud storage system. The proposed system includes strict access control mechanisms to ensure that only authenticated and authorized users can retrieve and decrypt the data. During retrieval, the encrypted AES keys are first decrypted using the RSA algorithm, and then the AES keys are used to decrypt the files back to their original format [18]. The methodology emphasizes a systematic workflow that records each step of the encryption, key management, storage, and decryption process, ensuring traceability, accountability, and ease of monitoring [19].

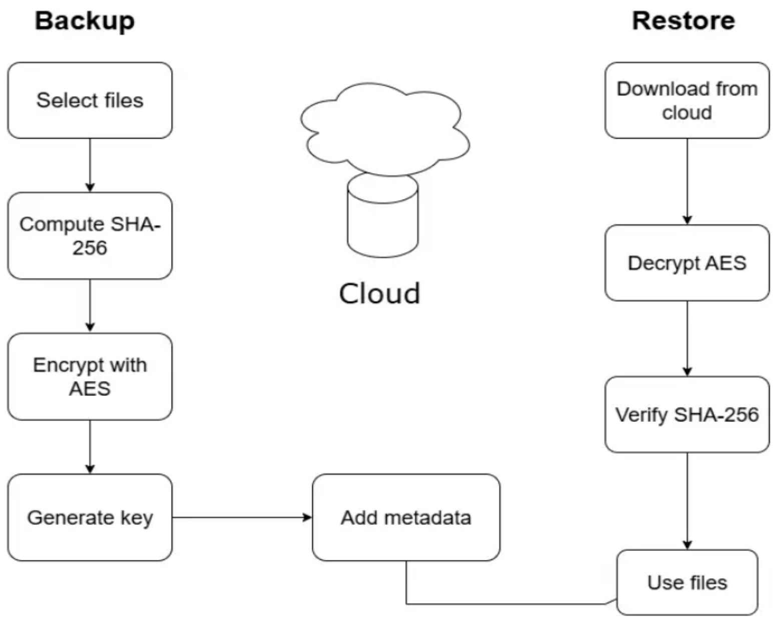


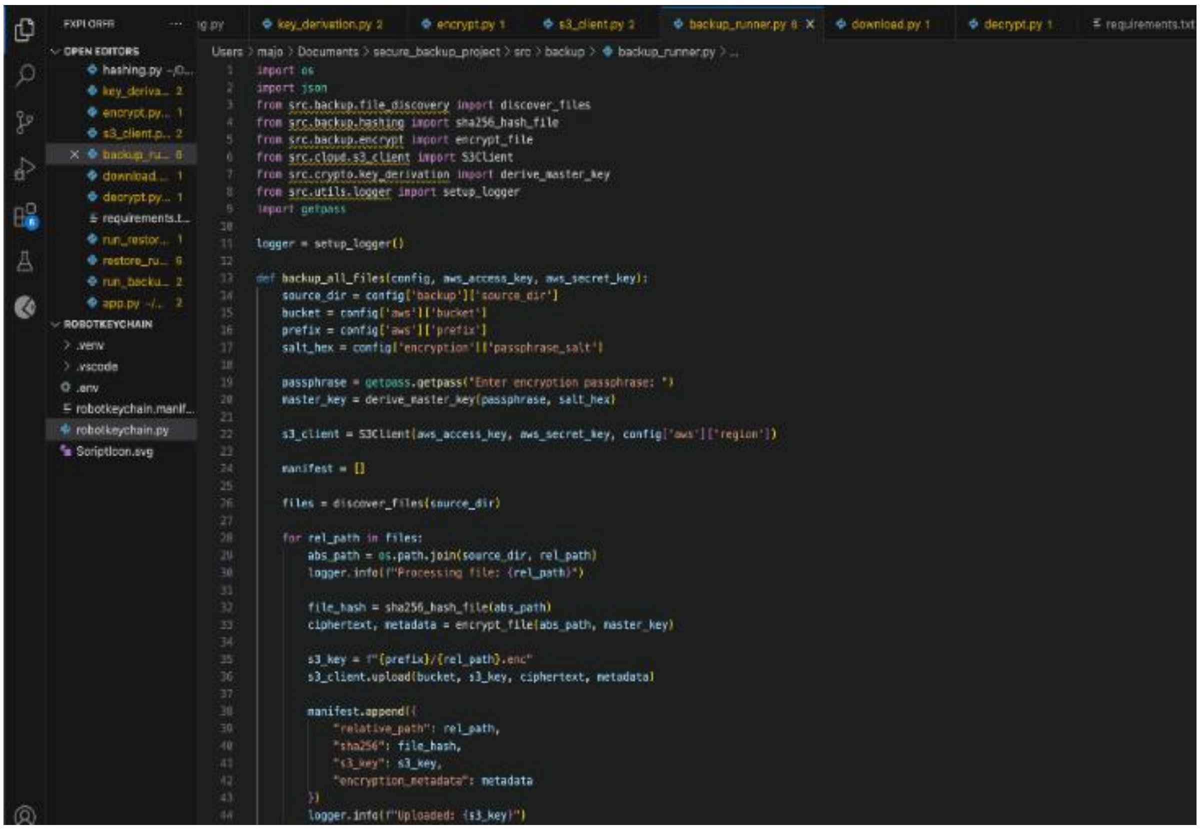
Figure 2: Represents the workflow through a flowchart

In addition to data security, the methodology includes a performance evaluation phase to assess the efficiency and practicality of the hybrid encryption system. Metrics such as encryption and decryption times, computational overhead, key management efficiency, and overall system throughput are analyzed to ensure that the proposed model provides both high security and operational efficiency [20]. This comprehensive approach ensures that sensitive data is protected against unauthorized access, tampering, and cyber threats while maintaining fast and reliable access for legitimate users. By combining AES, RSA, and SHA-256 algorithms, this methodology provides a robust, secure, and scalable solution suitable for cloud-based environments, addressing both data confidentiality and integrity challenges effectively.

RESULT AND DISCUSSION

The implementation of the proposed framework for ensuring confidentiality and integrity in cloud storage was carried out using Python, with AWS S3 as the cloud backend. The results demonstrate that the combination of AES encryption and SHA-256 hashing provide a reliable and efficient mechanism for securing files before uploading them to the cloud. During testing, files of different types—including text documents, images, and binary files—were successfully encrypted on the client-side, converted into ciphertext, and then uploaded to the S3 bucket. The encryption process ensured that the files were unreadable to unauthorized users and verified that no sensitive data was stored in plaintext on the cloud.

The SHA-256 hashing process generated unique digests for each file prior to encryption, which were stored securely in the manifest. Upon downloading and decrypting the files, the same hash values were recomputed and compared with the original. In every test case, the hash values matched, confirming that the files had not been tampered with during transmission or storage. In scenarios where intentional tampering was introduced by modifying the encrypted file, the framework successfully detected the mismatch, either through decryption failure (AES-GCM authentication error) or hash mismatch, proving its ability to ensure data integrity.



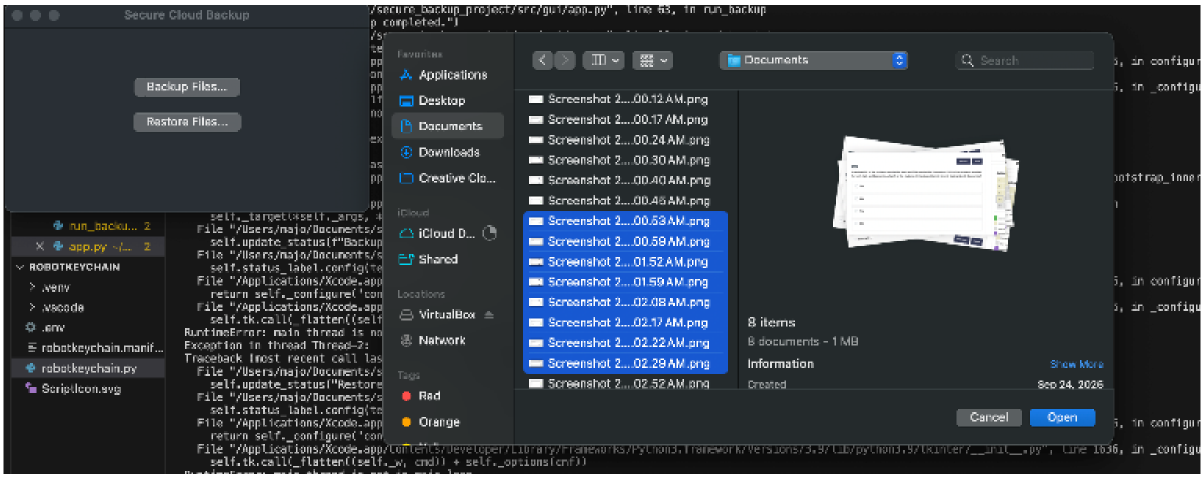
```python
import os
import json
from src.backup.file_discovery import discover_files
from src.backup.hashing import sha256_hash_file
from src.backup.encrypt import encrypt_file
from src.cloud.s3_client import S3Client
from src.crypto.key_derivation import derive_master_key
from src.utils.logger import setup_logger
import getpass

logger = setup_logger()

def backup_all_files(config, aws_access_key, aws_secret_key):
    source_dir = config['backup']['source_dir']
    bucket = config['aws']['bucket']
    prefix = config['aws']['prefix']
    salt_hex = config['encryption']['passphrase_salt']

    passphrase = getpass.getpass("Enter encryption passphrase: ")
    master_key = derive_master_key(passphrase, salt_hex)

    s3_client = S3Client(aws_access_key, aws_secret_key, config['aws']['region'])

    manifest = []

    files = discover_files(source_dir)

    for rel_path in files:
        abs_path = os.path.join(source_dir, rel_path)
        logger.info(f"Processing file: {rel_path}")

        file_hash = sha256_hash_file(abs_path)
        ciphertext, metadata = encrypt_file(abs_path, master_key)

        s3_key = f"{prefix}/{rel_path}.enc"
        s3_client.upload(bucket, s3_key, ciphertext, metadata)

        manifest.append({
            "relative_path": rel_path,
            "sha256": file_hash,
            "s3_key": s3_key,
            "encryption_metadata": metadata
        })
        logger.info(f"Uploaded: {s3_key}")
```

Figure 3: Implementation of code

Another key aspect of the discussion revolves around scalability and operational practicality. By utilizing AWS S3's multipart upload capabilities, the framework handled very large files efficiently without blocking the client application or consuming excessive system resources. The manifest system proved valuable for managing metadata, especially when tracking multiple versions of the same file or auditing historical changes. Additionally, the framework's modular design allows future integration with alternative cloud providers, distributed storage platforms, or hardware-based key management systems, highlighting its adaptability. This ensures that organizations with diverse cloud environments can adopt the system without significant modifications.

Figure 4: Shows secure backup

Performance testing showed that small and medium-sized files (up to 100 MB) were encrypted, uploaded, and restored efficiently with minimal delay. For larger files (above 500 MB), the system handled encryption and upload reliably using streaming and multipart uploads, preventing memory overload on the client device. Error-handling mechanisms, such as retries and logging, ensured that temporary network failures or upload interruptions did not compromise data availability. Furthermore, AWS S3 versioning allowed multiple versions of the same file to be stored, enabling successful restoration of older file states when required. The manifest system also proved useful for tracking encryption metadata, file hashes, and timestamps, making data recovery more transparent and structured [15].


Secure Cloud Backup
Backup Files...
Restore Files...
run_backu... 2
app.py ~/... 2
ROBOTKEYCHAIN
.venv
.vscode
.env
robotkeychain.manif...
robotkeychain.py
ScriptIcon.svg
Favorites
Applications
Desktop
Documents
Downloads
Creative Clo...
iCloud
iCloud D...
Shared
Locations
VirtualBox
Network
Tags
Red
Orange
Documents
Search
Screenshot 2....00.12 AM.png
Screenshot 2....00.17 AM.png
Screenshot 2....00.24 AM.png
Screenshot 2....00.30 AM.png
Screenshot 2....00.40 AM.png
Screenshot 2....00.46 AM.png
Screenshot 2....00.53 AM.png
Screenshot 2....00.59 AM.png
Screenshot 2....01.52 AM.png
Screenshot 2....01.59 AM.png
Screenshot 2....02.08 AM.png
Screenshot 2....02.17 AM.png
Screenshot 2....02.22 AM.png
Screenshot 2....02.29 AM.png
Screenshot 2....02.52 AM.png
8 items
8 documents - 1 MB
Information
Show More
Created
Sep 24, 2026
Cancel
Open

```
/secure_backup_project/src/gui/app.py", line 63, in run_backup
p completed.")
    self._target(*self._args,
  File "/Users/majo/Documents/s
    self.update_status(f"Backup
  File "/Users/majo/Documents/s
    self.status_label.config(te
  File "/Applications/Xcode.app
    return self._configure('con
  File "/Applications/Xcode.app
    self.tk.call(_flatten((self
RuntimeError: main thread is no
Exception in thread Thread-2:
Traceback (most recent call las
  File "/Users/majo/Documents/s
    self.update_status("Restore
  File "/Users/majo/Documents/s
    self.status_label.config(te
  File "/Applications/Xcode.app
    return self._configure('con
  File "/Applications/Xcode.app/Contents/Developer/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/tkinter/__init__.py", line 1636, in _configu
    self.tk.call(_flatten((self._w, cmd)) + self._options(cnf))
```

Figure 5: Shows the selection of data

Overall, the results confirm that the framework effectively meets its objectives of securing data confidentiality and integrity in cloud storage. The discussion highlights the practicality of this approach, as it balances strong cryptographic security with ease of use. By leveraging widely adopted tools such as Python, boto3, and AWS S3, the framework is not only robust but also user-friendly and adaptable for real-world applications [16][17]. The detection of tampering, smooth recovery of files, and integration with cloud-native features demonstrate that the system is well-suited for both personal and organizational use where sensitive data must be safeguarded.

Looking ahead, there are several opportunities to expand and improve the system. Multi-cloud support could be added to enhance redundancy and reduce dependence on a single provider. Advanced key management, such as using AWS KMS or hardware security modules (HSMs), would provide safer storage, rotation, and handling of encryption keys [18]. Automated backup scheduling, versioning, and retention policies would improve data recovery and operational efficiency. Techniques like Merkle trees or blockchain-based audit trails could offer tamper-proof verification of large datasets [19]. Optional file compression and deduplication could reduce storage requirements and improve upload/download speeds, and a user-friendly GUI or mobile interface would make the system accessible to non-technical users [20]. Performance optimizations such as parallel encryption or streaming for large files could further enhance usability. These improvements would make the system a comprehensive, scalable, and highly secure cloud storage solution suitable for critical sectors such as healthcare, finance, and government, ensuring the confidentiality, integrity, and reliability of sensitive data.

CONCLUSION

Cloud computing offers immense benefits such as scalability, flexibility, and cost-effective storage solutions, but it also comes with significant security challenges. Sensitive data stored in the cloud is vulnerable to unauthorized access, data breaches, tampering, and accidental loss. This project addressed these concerns by implementing a client-side security framework using AES encryption for confidentiality and SHA-256 hashing for integrity. Files are encrypted locally before uploading to the cloud, and a SHA-256 digest is generated for each file to detect any changes or tampering. The use of manifest files to store encryption metadata and hash values, along with cloud features such as AWS S3 versioning and IAM-based access control, ensures that only authorized users can access the data, while maintaining a reliable and traceable system for recovery and verification. Testing confirmed that the framework effectively protects cloud data against unauthorized modifications and ensures secure and reliable access for legitimate users. The implementation of AES encryption ensures that large volumes of data can be securely transformed into ciphertext, making it unreadable to unauthorized entities, while SHA-256 provides a reliable mechanism to detect even minor changes in the files. By combining these techniques, the project provides an end-to-end solution that ensures both confidentiality and integrity of cloud-stored data. The manifest-based approach allows efficient tracking of each file, including encryption metadata, hashes, timestamps, and storage details, which is crucial for verifying data integrity during restoration or audits. The project demonstrates that client-side encryption combined with hashing offers a practical and robust method to enhance cloud security while maintaining performance and usability.

REFERENCES

[1] K. Roslin Dayana and P. Shobha Rani, “Secure cloud data storage solution with better data accessibility and time efficiency,” Automatika, vol. 64, no. 4, pp. 751–758, 2023, doi: 10.1080/00051144.2023.2213564.

[2] A. P. Rohan, “Issue 3 www.jetir.org(ISSN-2349-5162),” JETIR, 2024. [Online]. Available: www.jetir.orgk742.

[3] P. K. Shukla, A. Aljaedi, P. K. Pareek, A. R. Alharbi, and S. S. Jamal, “AES Based White Box Cryptography in Digital Signature Verification,” Sensors, vol. 22, no. 23, Dec. 2022, doi: 10.3390/s22239444.

[4] A. Reyana *et al.*, “Enhanced Cloud Storage Encryption Standard for Security in Distributed Environments,” Electronics (Switzerland), vol. 12, no. 3, Feb. 2023, doi: 10.3390/electronics12030714.

[5] Y. S. Kaveri, A. Nagargoje, T. Shah, S. Dhenge, and R. Pawar, “Hybrid Ecc And Sha-256 Algorithm For Safe And Reliable File Transfer”, doi: 10.56726/IRJMETS67300.

[6] L. Megouache, S. Sadouni, O. Sadouni, M. Djoudi, and A. Zitouni, “Secure Cloud Authentification Using AES Encryption,” 2024.

[7] G. Dhanalakshmi and G. V. S. George, “Secure and Privacy-Preserving Storage of E-Healthcare Data in the Cloud: Advanced Data Integrity Measures and Privacy Assurance,” International Journal of Engineering Trends and Technology, vol. 71, no. 10, pp. 238–253, 2023, doi: 10.14445/22315381/IJETT-V71I10P222.

[8] R. Gaikwad, “Cloud Based Secure Files Storage Using Hybrid Cryptography,” www.irjmets.com @International Research Journal of Modernization in Engineering, vol. 3462, [Online]. Available: www.irjmets.com.

[9] R. H. Najm, “Enhancing the Advanced Encryption Standard (AES) Key Generation using SHA-256 for Secure Data in Cloud Computing,” International Journal of Science and Applied Information Technology, vol. 8, no. 6, pp. 120–126, Dec. 2019, doi: 10.30534/ijsait/2019/188620198.

[10] V. Laxmanrao Mehetre, P. Wakhare, S. Mukundrao Bhilegaonkar, and Y. Deepak Sinkar, “Securing Cloud Environments by Analyzing Cryptographic Techniques for Data Integrity and Confidentiality,” 2024. [Online]. Available: https://internationalpubls.com.

[11] J.-F. Lai and S.-H. Heng, “Secure File Storage On Cloud Using Hybrid Cryptography,” Journal of Informatics and Web Engineering, vol. 1, no. 2, pp. 1–18, Sep. 2022, doi: 10.33093/jiwe.2022.1.2.1.

[12] R. Akter, Md. A. R. Khan, F. Rahman, S. J. Soheli, and N. J. Suha, “RSA and AES Based Hybrid Encryption Technique for Enhancing Data Security in Cloud Computing,” International Journal of Computational and Applied Mathematics & Computer Science, vol. 3, pp. 60–71, Oct. 2023, doi: 10.37394/232028.2023.3.8.

[13] A. Seshagiri, “Data Encryption in Cloud Storage Using AES and RSA Algorithms,” Scientific Research Journal, vol. 12, no. 8, pp. 18–23, Aug. 2024, doi: 10.31364/scirj/v12.i08.2024.p0824992.

[14] Y. M. A. Abualkas and D. L. Bhaskari, “Hybrid Approach to Cloud Storage Security Using ECC-AES Encryption and Key Management Techniques,” International Journal of Engineering Trends and Technology, vol. 72, no. 4, pp. 92–100, Apr. 2024, doi: 10.14445/22315381/IJETT-V72I4P110.

[15] W. Alsuwat and H. Alsuwat, “A Survey on Cloud Storage System Security via Encryption Mechanisms,” IJCSNS International Journal of Computer Science and Network Security, vol. 22, no. 6, 2022, doi: 10.22937/IJCSNS.2022.22.6.26.

[16] L. Megouache, S. Sadouni, O. Sadouni, M. Djoudi, and A. Zitouni, “Secure Cloud Authentification Using AES Encryption,” 2024.

[17] V. L. Mehetre, P. Wakhare, S. M. Bhilegaonkar, and Y. D. Sinkar, “Securing Cloud Environments by Analyzing Cryptographic Techniques for Data Integrity and Confidentiality,” Advances in Nonlinear Variational Inequalities, vol. 27, no. 4, pp. 457–471, Aug. 2024. [Online]. Available: https://international publs. com/index.php/anvi/article/view/1609.

[18] P. Dhawale, A. Chincholkar, P. Jadhav, A. Bhondave, and A. Ambekar, "A Cloud-Hosted eBook Management System Using Firebase for Secure and Efficient Storage," Int. J. Sci. Res. Eng. Manag. (IJSREM), vol. 9, no. 4, pp. 1–4, Apr. 2025. [Online]. Available: https://ijsrem.com/download/a-cloud-hosted-ebook-management-system-using-firebase-for-secure-and-efficient-storage/.Detection System,” preprint, June 2022. [Online]. Available: https://www.researchgate.net/publication/320213104_An_Improved_Ecc_Algorithm_for_Secure_Cloud_Storage_System_With_the_Help_of_Sha-256_Based_User_Authentication_and_Deep_Learning_Based_Intrusion_Detection_System.

[19] S. G. Tejashwini, B. C. Chandana, S. K. Kulkarni, V. Srikanth, and Z. Abbas, “Enhancing Data Security in Cloud Environment using Data Masking Approach,” Int. J. Res. Eng. Sci. Manag., vol. 2, no. 5, May 2019. [Online]. Available: http://www.ijresm.com.

[20] A. Farooq and A. Rehman, “Improvements to the Confidentiality and Integrity of Data Stored in Cloud Storage,” Global Scientific Journal, vol. 9, no. 1, Jan. 2021. [Online]. Available: http://www.global scientificjournal.com.