IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151172

Simple Sovereign Cloud Storage

Sudarshan SR[1], Jai Kaarthik B[2], Durgadevi. P[3]

Student, Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[1]

Student, Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[2]

Assistant Professor, Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[3]

Abstract: Cloud storage has become an essential backbone for personal and organizational data management in the modern digital era. However, existing commercial cloud architectures such as Amazon Web Services (AWS), Google Cloud, Dropbox, and Microsoft Azure rely heavily on third-party hosting, where external technology corporations control the physical servers and security infrastructure. This dependence introduces significant limitations for users, including a lack of data sovereignty, recurring subscription and hidden operational costs, and vulnerability to third-party service outages. Furthermore, while standard hybrid cryptography models combine the Advanced Encryption Standard (AES) for rapid payload encryption with Rivest-Shamir-Adleman (RSA) for secure key transport, they suffer from a critical blind spot: the absence of data integrity verification. If an attacker intercepts an encrypted file and flips random bits, or if the file becomes corrupted over an unstable network connection, a standard AES and RSA setup provides no warning and decrypts a corrupted or maliciously altered file silently.

Simple Sovereign Cloud Storage (SSCS) is a fully sovereign, globally accessible cloud storage environment hosted entirely on local hardware to eliminate reliance on third-party vendors. The system exposes a local hard drive storage medium as a distributed file system using Network File Share (NFS) on a Unix-based operating system, guarded by a customized firewall (such as firewalld or ufw) to restrict unauthorized and malicious network traffic. Client devices interact with the cloud server through a command line interface (CLI) that performs encryption and decryption. It uses AES to encrypt the file data, and Secure Hash Algorithm 256-bit (SHA-256) generates and verifies cryptographic digests to guarantee end-to-end data integrity.

Keywords: Sovereign Cloud Storage, Advanced Encryption Standard (AES), Data Integrity Verification, Network File System (NFS)

I. INTRODUCTION

This paper gives an overview about the aim, objectives, background and operation environment of the system.

The aim of this paper is to provide:

- A fully sovereign, self-hosted cloud file storage environment hosted on local hardware without third-party control
- Fast and confidential file encryption using AES
- End-to-end data integrity verification using SHA-256 hashing to detect bit-flipping, tampering, or network corruption
- Distributed file access via Network File Share (NFS) protected by a customized firewall and an easy-to-use GUI Client Interface.

II. BACKGROUND

The Client Cryptographic Interface is an application layer through which end users (such as Client #1 and Client #2) interact with the storage system. Before any file leaves the client device, it is processed through the hybrid cryptographic engine, which computes a SHA-256 hash for integrity, and encrypts the file payload using AES.

The Sovereign Cloud Server is a Unix-based backend infrastructure that hosts the physical hard drive storage medium via Network File Share (NFS). Incoming upload requests and outgoing download streams pass through a customized firewall (firewalld/ufw) that filters out unwanted or malicious traffic before reaching the NFS storage layer.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151172

The functionality provided by these two components allows small-scale and personal users to maintain complete ownership over their files, avoid recurring subscription fees, minimize downtime, and guarantee both confidentiality and integrity of stored data.

III.SYSTEM ANALYSIS

The following section discusses the development process of **Simple Sovereign Cloud Storage (SSCS)** including the software requirement specification (SRS) and a comparison between the existing system in place and the proposed system. The various functional and non-functional requirements are included in the SRS to ensure that a complete overview of the system is prepared before the beginning of the development process. The comparison with the existing system in place provides a view of how the proposed system aims to resolve the issues in current commercial cloud architectures and standard hybrid cryptography models in an efficient manner.

Table I Minimum System Requirements

Processor	Any x86 / x86_64 processor
Operating System	Unix-based OS (Server), Windows / Linux / macOS (Client)
Memory	2 GB RAM (minimum)
Hard Drive Storage	Dedicated Hard Drive to act as storage medium
Network & Security	Network Router, Firewall (firewalld / ufw)
File System & Crypto	Network File Share (NFS), AES, SHA-256

IV. SOFTWARE REQUIREMENT SPECIFICATION

A. General Description

1) Product Description:

Simple Sovereign Cloud Storage (SSCS) is a self-hosted, sovereign distributed file storage system that enables users to store and retrieve files securely over a network using local hardware. It combines Network File Share (NFS) and firewall traffic filtering with a two-layer cryptographic scheme (AES, and SHA-256) to eliminate third-party data exposure, recurring subscription costs, and silent file tampering.

2) Problems Aimed To Solve:

The various problems in existing commercial cloud architectures and standard hybrid encryption systems include:

- Lack of Data Sovereignty:
  - Existing commercial cloud providers (such as AWS, Google Cloud, Dropbox, and Microsoft Azure) store user files on third-party physical servers, meaning users surrender direct physical control over sensitive data.
- Hidden Costs and Service Outages:
  - Commercial cloud platforms impose recurring subscription fees, bandwidth charges, and vendor lock-in, while centralized server outages can temporarily block access to personal or organizational files.
- Lack of Data Integrity in Standard AES + RSA Models:
  - While AES quickly encrypts file data and RSA safely transports the AES secret key, this combination only covers confidentiality. If a hacker intercepts an encrypted file and flips random bits, or if the file gets corrupted over a bad network connection, the system does not warn the user and decrypts a broken or maliciously altered file.
- Unfiltered Network Exposure:
  - Exposing a file server over the internet without strict host-level packet filtering leaves storage ports vulnerable to unauthorized probing and malicious traffic.
  - The proposed system aims to provide a solution for each of the problems mentioned above through a locally hosted Unix/NFS cloud server protected by a customized firewall, paired with client-side AES encryption, and SHA-256 integrity verification.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151172

V. SYSTEM REQUIREMENTS

A. *Non-Functional Requirements*

1) *Usability Requirement:*
The system should provide a simple and intuitive User Interface so that clients can easily upload, encrypt, download, and decrypt files without complex command-line operations.

2) *Security and Confidentiality Requirement:*
The system must ensure that all files are encrypted using AES prior to transmission, and unwanted network traffic is blocked by the customized firewall.

3) *Data Integrity Requirement:*
The system must compute and verify SHA-256 cryptographic hashes for every stored file so that any bit-flipping, network corruption, or malicious modification is immediately detected prior to usage.

4) *Availability and Sovereignty Requirement:*
The system should be hosted entirely on local hardware with a simple and efficient setup that provides immediate usage, minimizes downtime, and eliminates recurring subscription fees.

B. *Functional Requirements*

The system should be able to provide the following features across its core operational layers:

1) *Client Users:*
   a) This role is assigned to authorized end users interacting with the system from internet-enabled client devices.
   b) Users should be able to select files for upload and initiate encryption and storage through the User Interface.
   c) Users should be able to request stored files from the cloud server, verify their SHA-256 integrity, and decrypt them locally.

2) *Cryptographic Engine (Client Interface):*
   a) This module handles the Encrypt and Decrypt operations between the Client Interface and the Cloud Server.
   b) During upload, it computes the SHA-256 hash of the original file, encrypts the file payload using AES.
   c) During download, it decrypts the AES ciphertext, and recomputes the SHA-256 digest to warn the user if any corruption or tampering occurred.

3) *Cloud Server, Firewall, and NFS Layer:*
   a) The customized firewall inspects all incoming and outgoing packets to block unauthorized or malicious traffic before it reaches the storage service.
   b) The Network File Share (NFS) service exposes the physical hard drive storage medium as a distributed file system to store and retrieve encrypted data blocks.

VI. HARDWARE REQUIREMENTS

- **A Hard Drive** to act as the primary physical storage medium for encrypted files
- A device with internet access to host the **Unix-based cloud server**
- **A Network Router** to route incoming and outgoing traffic between clients and the server
- Internet-enabled devices which act as **clients** to interact with the server

A. *Software Requirements*

- Network File Share (NFS) to expose the storage medium as a distributed file system.
- User Interface to allow users to access the server and trigger encryption/decryption
- A Firewall, such as firewalld or ufw, to restrict incoming and outgoing network traffic
- A Unix-based operating system to act as the server platform.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151172

VII. SYSTEM COMPARISON

A. Existing System

- Existing commercial cloud architectures (Amazon Web Services, Google Cloud, Dropbox, Microsoft Azure) rely heavily on third-party hosting, leading to a lack of data sovereignty, hidden subscription costs, and vulnerability to service outages.
- Existing hybrid cryptography models rely on standard AES + RSA, which only covers confidentiality and lacks data integrity; if a hacker flips random bits or a file is corrupted over a bad network connection, the system decrypts the corrupted file without warning.

B. Proposed System

- Instead of trusting third-party vendors with sensitive data, we build a fully sovereign, globally accessible cloud environment hosted entirely on our own local hardware.
- Data is never held by a third-party server, and a simple, efficient setup provides immediate usage, minimizes downtime, and incurs no recurring subscription fees.
- Usage of AES provides fast symmetric encryption of file data.
- Usage of SHA-256 ensures end-to-end data integrity, and deployment of a customized firewall prevents unwanted or malicious traffic.

C. Software Used

- Unix-based Operating System is used as the core host environment for the sovereign cloud server due to its stability, native networking stack, and granular permission model.
- Network File Share (NFS) protocol is used to mount and expose the local hard drive storage medium as a distributed file system accessible across the network.
- Firewall utilities (such as firewalld or ufw) are used to configure packet filtering rules that restrict incoming and outgoing traffic strictly to authorized client requests.
- Client Application is used on the client side to provide seamless file upload and download capabilities for multiple users (Client #1, Client #2).
- Hybrid Cryptographic Suite (AES + SHA-256) is integrated into the client workflow, where Advanced Encryption Standard (AES) performs fast symmetric encryption of bulk file payloads, and Secure Hash Algorithm 256-bit (SHA-256) computes cryptographic digests for integrity verification.

VIII.SYSTEM DESIGN

A. System Architecture

The system architecture is structured into three distinct operational tiers: the Users Tier, the Client Interface Tier, and the Cloud Server Tier.

1) Users Tier:

a) Represents the end users operating internet-connected client devices who initiate file upload and download operations.

2) Client Interface Tier (Interface & Encrypt / Decrypt Boundary):

a) All clients connect using a dedicated interface.

b) Before data crosses from the Client Interface to the Cloud Server, it passes through the Encrypt / Decrypt cryptographic boundary, where SHA-256 hashing, AES payload encryption/decryption.are performed.

3) Cloud Server Tier (Firewall, NFS, and Data Storage):

a) All encrypted traffic entering or leaving the Cloud Server first passes through the customized Firewall.

b) Validated requests are forwarded to the Network File Share (NFS) service, which reads from and writes to the physical Data (Hard Drive) storage medium.

B. Use Case Description

- Client User
- Selects local files via the Interface, encrypts them using AES, and uploads them to the NFS Cloud Server.
- Requests stored files from the Cloud Server, verifies the SHA-256 digest to confirm no tampering or corruption occurred, and decrypts the file locally.
- Server Administrator

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151172

- Configures the Unix server, manages NFS shared directories on the hard drive, and updates firewalld/ufw rules to block unauthorized traffic.

C. Sequence and Data Workflow

1) File Upload and Encryption Sequence:

a) The user selects a file inside the Interface.

b) Upon first time usage, a random AES key is generated. This is used to encrypt the file before transmission.

c) The cryptographic module computes the SHA-256 digest of the file to establish a baseline integrity fingerprint.

d) The encrypted payload alone is transmitted across the network router to the Cloud Server.

e) The customized Firewall inspects the incoming packets; if authorized, it forwards the stream to the Network File Share (NFS) service, which writes the encrypted data to the Hard Drive.

2) File Download and Decryption Sequence:

a) The user requests a stored file through the Interface.

b) The Firewall validates the request and retrieves the encrypted file package from the Hard Drive via the NFS service.

c) Upon receiving the package at the Client Interface, the SHA-256 integrity of the file is verified. If the hashes don't match, the file is marked as corrupted and a warning is displayed.

d) Once verified, the payload is decrypted using the private AES key.

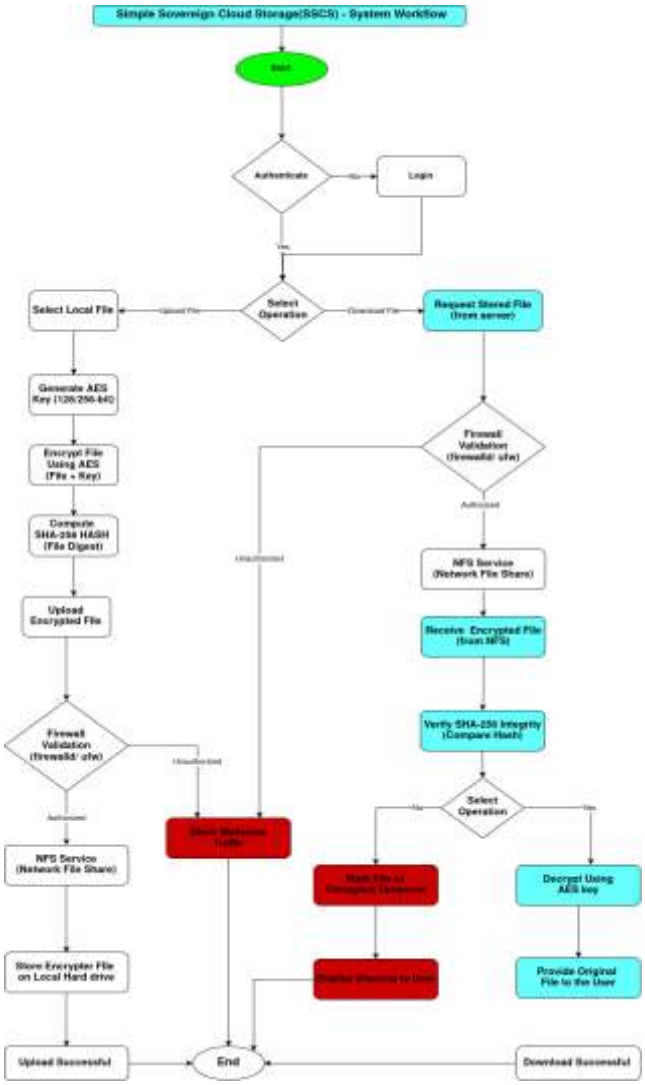


Figure I. Flowchart of the Proposed Simple Sovereign Cloud Storage System



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151172

D. Cryptographic Methods Used

1) The system combines multiple cryptographic and software engineering paradigms to build a secure, flexible, and extensible cloud storage framework:
   a) Symmetric Key Cryptography (AES – Advanced Encryption Standard) for high-speed block encryption of large file payloads
   b) Cryptographic Hash Functions (SHA-256 – Secure Hash Algorithm 256-bit) for tamper detection and end-to-end data integrity verification
   c) Distributed File System Architecture (NFS – Network File Share) for exposing local physical storage over a network.

IX. SYSTEM IMPLEMENTATION

A. Sovereign Cloud Server & NFS Configuration

- A Unix-based operating system is deployed on the dedicated local server hardware connected to the internet via a network router.
- The physical hard drive storage medium is formatted and mounted to a dedicated storage directory on the server.
- Network File Share (NFS) is configured via export rules to expose the storage directory as a distributed file system accessible only to authenticated client interfaces.

B. Customized Firewall Deployment:

- A host-level firewall (firewalld / ufw) is enabled on the Unix server to act as the security gatekeeper between the external network and the NFS service.
- Default deny policies are applied to all incoming traffic, and explicit allow rules are configured strictly for the NFS and client communication ports from trusted client IP ranges.
- Unwanted probing, port scanning, and malicious traffic packets are dropped at the firewall boundary before they can reach the NFS daemon or physical data drive.

C. Hybrid Encryption & Integrity Module

- AES Encryption Engine: Reads the raw byte stream of the user's file and encrypts it using a private AES key, ensuring high throughput and low latency even for large files.
- SHA-256 Integrity Verifier: Generates a 256-bit cryptographic hash of the file prior to upload and verifies it upon download. Any single-bit alteration during transit or storage produces a completely different hash value, immediately flagging data corruption or active tampering.

D. Client Interface

- Provides a clear interface for Client #1 and Client #2 to browse local files, trigger encrypted uploads to the NFS mount, view stored files on the cloud server, and execute verified downloads.

X.TESTING

- To test the functionality, security, and reliability of the developed system, multiple client environments (Client #1 and Client #2) were connected to the Unix-based NFS cloud server through the Interface.
- Files of varying formats and sizes (documents, archives, and media files) were uploaded through the Interface. Packet inspection confirmed that all file payloads stored on the NFS hard drive were fully encrypted via AES.
- To verify the core improvement over standard AES systems, encrypted files stored on the server were intentionally modified by flipping random bits to simulate network corruption and man-in-the-middle tampering. Upon download, the SHA-256 verification module successfully detected the hash mismatch and alerted the user instead of silently outputting a corrupted file.
- Unauthorized connection attempts and simulated malicious traffic from external unapproved IP addresses were directed at the server. The customized firewall (firewalld / ufw) successfully blocked all unauthorized requests while permitting legitimate encrypted transfers from Client #1 and Client #2.

XI.CONCLUSION & SCOPE OF THE EXTENSION

- Deployment of Secure Cloud File Storage using Hybrid Encryption provides a fully sovereign, self-hosted cloud storage environment on local hardware without relying on third-party commercial vendors such as AWS, Google Cloud, Dropbox, or Microsoft Azure.
- The system eliminates recurring subscription fees, minimizes setup downtime, and ensures complete ownership and privacy of personal or small-scale organizational data.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151172

- By integrating SHA-256 hashing alongside AES data encryption, the system resolves the critical integrity blind spot of traditional hybrid cryptography and protects users against silent file corruption and malicious bit-flipping.
- The combination of Network File Share (NFS) and a customized firewall ensures fast distributed file access while blocking unwanted and malicious network traffic.
- For future extension, the system can be expanded with a mobile client application, automated periodic AES key rotation, and multi-drive RAID redundancy for automated local hardware fault tolerance.

REFERENCES

[1]. V. Sharma, A. Chauhan, H. Saxena, S. Mishra and S. Bansal, "Secure File Storage on Cloud using Hybrid Cryptography," 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-6, doi: 10.1109/ISCON52037.2021.9702323.
[2]. National Institute of Standards and Technology. (2001, updated May 9, 2023). Advanced Encryption Standard (AES). NIST FIPS 197-upd1.
[3]. National Institute of Standards and Technology. Secure Hash Standard (SHS). Federal Information Processing Standards Publication (NIST FIPS) 180-4. Gaithersburg, MD: National Institute of Standards and Technology, 2015. https://doi.org/10.6028/NIST.FIPS.180-4. [1, 2]
[4]. S. Shepler, M. Eisler, and D. Noveck, Eds., "Network File System (NFS) Version 4 Protocol," RFC 7530, Mar. 2015. [Online]. Available: https://dl.acm.org/doi/10.17487/RFC7530
[5]. https://firewalld.org/documentation/

 This work is licensed under a Creative Commons Attribution 4.0 International License