

Comparative Analysis of Reed-Solomon Configurations in a Secure Distributed S3 Storage Framework

Trapti Nagwanshi¹, Dr. Ruchika Pachori²

Department of Information Technology, Mahakal Institute of Technology, Ujjain, Madhya Pradesh, India

ABSTRACT

Managing large-scale data in distributed environments has become increasingly challenging, particularly when organizations require both high availability and cost efficiency. Conventional replication-based storage systems address fault tolerance by maintaining multiple identical copies of data, but this approach introduces substantial storage overhead, making it less economical at scale.

This paper builds on a previously published secure and storage-efficient framework for distributed S3-compatible clusters, which integrates Reed-Solomon erasure coding with a client-side cryptographic pipeline consisting of AES-256-GCM encryption and SHA-256 integrity hashing in a MinIO-based environment. While the earlier study evaluated a fixed (4,2) configuration, the present work extends the framework by examining three Reed-Solomon configurations—(4,2), (6,2), and (8,4)—to investigate their storage-efficiency, fault-tolerance, and performance characteristics while retaining the same client-side security architecture.

The configuration analysis shows that (6,2) provides the lowest storage factor among the evaluated configurations (1.33x), with tolerance for two simultaneous shard failures out of eight (25%). In comparison, (8,4) retains the 1.50x storage factor of the (4,2) baseline while increasing the absolute number of tolerable simultaneous shard failures from two to four. In the controlled client-side benchmark, the measured total processing time for a 1 GB input file was 6.597 s for (4,2), 5.942 s for (6,2), and 9.398 s for (8,4). Reconstruction verification succeeded byte-for-byte in every reported trial. The previously published distributed-testbed measurements for (4,2) are retained separately as a historical reference baseline.

The contribution of this study is a configuration-level extension of the established framework. The underlying security technology and storage architecture remain unchanged, while the Reed-Solomon data-to-parity configuration is treated as the primary variable for analyzing storage efficiency, fault tolerance, and processing behavior.

Keywords: *Distributed S3 Storage, Reed-Solomon Erasure Coding, Configuration Analysis, AES-256-GCM Encryption, SHA-256 Integrity Verification, MinIO, Storage Efficiency, Fault Tolerance, Performance Evaluation.*

I. INTRODUCTION

The rapid growth of digital services, cloud applications, and data-intensive workloads has significantly increased the

demand for reliable and cost-effective storage infrastructure. Organizations require storage systems that provide high availability, data durability, efficient resource utilization, and protection against unauthorized access. Distributed S3-compatible object storage clusters have emerged as a practical solution for managing large volumes of unstructured data. However, achieving an appropriate balance between storage efficiency, fault tolerance, and data security remains a major design challenge [2], [4].

Traditional distributed storage systems commonly use replication to maintain data availability. In a triple-replication model, three identical copies of each data object are stored across different nodes. Although this approach provides redundancy, it requires approximately three times the original data capacity, resulting in a 200% storage overhead. As data volumes increase, this additional capacity can substantially raise infrastructure costs and the total cost of ownership [3], [4]. Consequently, erasure coding has gained attention as an alternative redundancy mechanism that can provide fault tolerance with lower storage overhead.

Reed-Solomon (RS) erasure coding divides data into multiple data shards and generates additional parity shards. These parity shards enable data reconstruction when a specified number of shards become unavailable or corrupted. A configuration represented by (k,m) consists of k data shards and m parity shards, with a total of k+m shards. Under the erasure-coding model, the system can tolerate the loss of up to m shards, provided that the remaining shards are available for reconstruction. The storage factor is calculated as (k+m)/k, making the selection of data and parity shard counts an important consideration in distributed storage design.

In addition to storage efficiency and fault tolerance, data confidentiality and integrity are essential requirements for cloud-based storage systems. Client-side encryption helps protect data before it is transferred to the storage provider, while cryptographic hashing can be used to verify data integrity during retrieval. The previously published framework integrated AES-256-GCM encryption and SHA-256 integrity hashing with Reed-Solomon erasure coding in a MinIO-based S3-compatible environment. That study evaluated a fixed (4,2) configuration and demonstrated the

feasibility of combining client-side security with storage-efficient data distribution [1].

However, the earlier framework did not provide a direct experimental comparison of alternative Reed-Solomon configurations under a common processing environment. The selection of an erasure-coding configuration can influence storage overhead, the number of tolerable shard failures, encoding and reconstruction costs, and overall processing behavior. Therefore, evaluating multiple configurations is necessary to understand the trade-offs associated with different data-to-parity arrangements.

To address this gap, the present study extends the previously published framework by evaluating three Reed-Solomon configurations: (4,2), (6,2), and (8,4). The AES-256-GCM encryption, SHA-256 integrity verification, and MinIO-based storage architecture are retained, while the Reed-Solomon configuration is treated as the primary variable for comparative analysis. The study examines storage factor, fault-tolerance capacity, processing time, and reconstruction behavior under a controlled client-side benchmarking environment. The previously published distributed-testbed measurements are maintained separately as a historical reference and are not combined with the controlled benchmark results.

The main contributions of this study are as follows:

1. A comparative evaluation of three Reed-Solomon configurations—(4,2), (6,2), and (8,4)—within an established secure distributed storage framework.
2. An analysis of the relationship between data-to-parity configuration, storage factor, and theoretical shard-failure tolerance.
3. A controlled assessment of processing and reconstruction behavior while retaining the same client-side AES-256-GCM and SHA-256 security pipeline.
4. A configuration-level analysis that provides measurable evidence for understanding storage-efficiency and performance trade-offs without introducing a new storage or security architecture.

II. RELATED WORK

Distributed storage has become an important area of research as the volume of digital information continues to grow. One of the major challenges in these systems is maintaining data availability without consuming excessive storage resources. Replication is a commonly used approach in which multiple copies of the same data are stored across different nodes. While this method provides redundancy, it also increases storage requirements and operational costs. These limitations have motivated researchers to explore erasure coding as an alternative for achieving fault tolerance with reduced storage overhead [2], [3].

Reed-Solomon coding is widely used in storage systems because it allows lost data to be reconstructed from the available data and parity shards. The technique provides flexibility in selecting the number of data and parity shards according to system requirements. However, different configurations introduce different trade-offs in

terms of storage consumption, failure tolerance, and computational complexity. Previous studies on erasure coding have examined these characteristics and highlighted the importance of selecting an appropriate configuration for a given storage environment [3], [7], [14].

The development of cloud-native storage platforms has further increased interest in distributed object storage. MinIO provides an S3-compatible environment in which erasure coding is used to support data protection and recovery across storage drives. Its deployment documentation describes how data and parity shards contribute to storage availability and fault tolerance. These capabilities make configuration selection an important consideration when designing storage clusters for different workloads [5], [12].

Data security is another essential aspect of cloud storage. Encryption helps protect confidential information from unauthorized access, while cryptographic integrity mechanisms are used to detect unintended changes to stored data. AES is a standardized encryption algorithm, and its authenticated encryption variants can provide confidentiality along with authentication. Research on cryptographic cloud storage and encryption overhead has also examined the relationship between security mechanisms and system performance [1], [8], [10], [13]. Therefore, evaluating storage efficiency without considering the processing cost of security operations may provide an incomplete picture of system performance.

In addition to storage and security, the computational requirements of erasure coding have received attention in research. Reed-Solomon encoding relies on arithmetic operations over finite fields, and the efficiency of these operations can influence encoding and reconstruction time. Similarly, the performance of distributed storage depends on factors such as data size, storage configuration, and the underlying execution environment. These considerations highlight the need to evaluate storage configurations under consistent experimental conditions [6], [7], [9], [11].

Despite the progress in these areas, the relationship between Reed-Solomon configuration, storage efficiency, fault tolerance, and processing behavior remains an important consideration in the design of secure distributed storage systems. A configuration that reduces storage overhead may introduce different computational requirements, while a configuration with additional parity shards may provide greater theoretical failure tolerance at a different storage cost. These trade-offs need to be examined together rather than in isolation.

The present study investigates these aspects by comparing three Reed-Solomon configurations—(4,2), (6,2), and (8,4)—within a common secure distributed S3 storage framework. The study retains the same client-side encryption, integrity verification, and storage architecture while varying the data-to-parity

configuration. This enables a focused comparison of storage factor, theoretical fault tolerance, processing time, and reconstruction behavior without introducing changes to the underlying security framework.

III. PROPOSED SYSTEM AND ARCHITECTURE

3.1 Overview of the Proposed Framework

The proposed framework is designed to examine the performance of different Reed-Solomon configurations in a secure distributed S3-compatible storage environment. Distributed storage systems employ redundancy mechanisms to maintain data availability and reliability, while erasure coding provides an alternative to conventional replication by reducing storage overhead [3], [9]. The framework integrates client-side encryption, integrity verification, erasure coding, and distributed object storage to provide a structured approach to data protection and storage management.

AES-256-GCM is employed to encrypt data before it is transferred to the storage layer, while SHA-256 hashing supports integrity verification during data processing and retrieval [4], [13]. Following encryption, the data is divided into smaller shards, and parity information is generated using Reed-Solomon erasure coding [14]. These shards are subsequently managed through MinIO, an S3-compatible object storage platform that supports erasure-coded data protection [12].

The primary objective of this study is to evaluate how different data-to-parity configurations influence storage efficiency, fault tolerance, and processing time while retaining the same security mechanisms and storage architecture. Three configurations, namely (4,2), (6,2), and (8,4), are considered for comparative analysis.

3.2 System Architecture

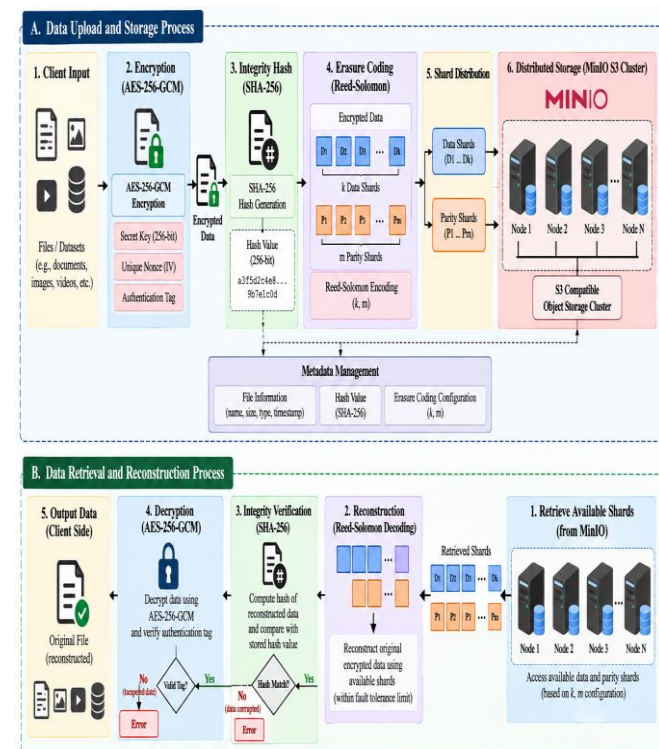


Fig. 1. Architecture of the proposed secure distributed S3 storage framework.

The proposed system architecture integrates client-side encryption, data integrity verification, Reed-Solomon erasure coding, and distributed object storage into a unified framework. It is designed to provide data confidentiality, fault tolerance, and reliable file reconstruction in an S3-compatible storage environment. The use of erasure coding enables data recovery from available shards while reducing the storage overhead associated with conventional replication techniques [3], [9]. The overall architecture, illustrated in Fig. 1, consists of two major operations: the data upload and storage process, and the data retrieval and reconstruction process.

During the upload process, the input file is first encrypted using AES-256-GCM on the client side, ensuring that the original data is protected before being transferred to the storage environment [10], [13]. A SHA-256 hash is generated to support subsequent integrity verification. The encrypted data is then processed using Reed-Solomon erasure coding, which divides it into data and parity shards according to the selected configuration [14]. These shards are distributed across multiple nodes in the MinIO S3-compatible storage cluster, following the principles of distributed object storage and erasure-coded data placement [12]. Relevant metadata, including file information, hash values, and erasure-coding parameters, is maintained to support future retrieval and reconstruction.

During the retrieval process, the system accesses the available shards from the distributed storage cluster and uses Reed-Solomon decoding to reconstruct the

encrypted file. The reconstructed data is subjected to SHA-256 integrity verification by comparing its computed hash with the stored reference value. If the integrity check is successful, AES-256-GCM decryption is performed, including authentication-tag verification, to recover the original file. This architecture combines cryptographic protection with erasure coding to support secure storage and recovery in the presence of node or shard failures within the configured fault-tolerance limit [3], [10], [12].

3.3 Client-Side Encryption Using AES-256-GCM

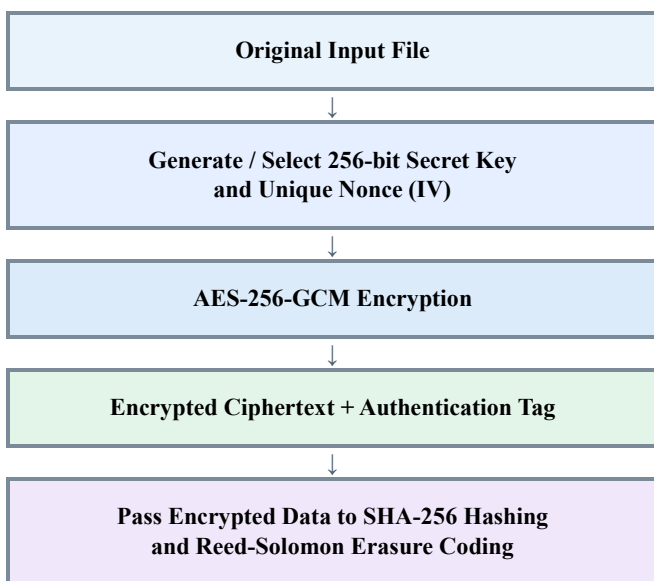


Fig. 2. Client-side AES-256-GCM encryption workflow.

Client-side encryption is incorporated into the proposed framework to protect data before it enters the distributed storage environment. The Advanced Encryption Standard (AES) is a symmetric-key encryption algorithm widely used to provide confidentiality for digital information [13]. In this framework, AES-256 is combined with Galois/Counter Mode (GCM), which provides both data encryption and authentication within a single cryptographic operation [15].

Before encryption, the input file is processed using a secret key and a unique initialization vector (IV), also referred to as a nonce. The AES-256-GCM algorithm transforms the original data into ciphertext and generates an authentication tag. The ciphertext conceals the original content, while the authentication tag enables verification that the encrypted data has not been altered. The uniqueness of the IV for a given key is an important requirement for maintaining the security of GCM [15].

Encrypting data at the client side reduces the exposure of plaintext within the storage infrastructure because the data is protected before transmission to the MinIO cluster. The resulting ciphertext is subsequently passed to the integrity-verification and Reed-Solomon encoding stages, where it is prepared for distributed storage. This arrangement separates cryptographic protection from the underlying storage

mechanism and allows the framework to combine confidentiality with the fault-tolerance capabilities of erasure coding [10], [12].

The encryption process is illustrated in Fig. 2. It represents the transformation of the original input file into authenticated ciphertext before the data proceeds to the subsequent stages of the proposed storage framework.

3.4 SHA-256 Integrity Verification

Data integrity is an essential component of secure distributed storage, as data may be exposed to accidental corruption or unauthorized modification during storage and retrieval. To address this concern, the proposed framework incorporates the Secure Hash Algorithm 256 (SHA-256) to verify the integrity of stored data. SHA-256 is a cryptographic hash function that generates a fixed-length 256-bit digest from input data, making it suitable for detecting changes in stored information [4].

In the proposed framework, the SHA-256 digest is computed over the encrypted file before it is divided into shards for distributed storage. The generated digest is retained as a reference value along with the relevant file metadata. During retrieval, the available shards are first reconstructed using Reed-Solomon decoding. The SHA-256 digest of the reconstructed ciphertext is then calculated and compared with the previously stored reference digest. If both values match, the reconstructed data is considered consistent with the originally stored ciphertext. A mismatch indicates that the data may have been modified or corrupted and requires further investigation.

The integrity-verification mechanism operates independently of the encryption process and complements the authentication capabilities of AES-256-GCM. While SHA-256 provides a

Configuration	Data Shards (k)	Parity Shards (m)	Total Shards	Storage Factor	Failure Tolerance
RS (4,2)	4	2	6	1.50×	2 shards (33.3%)
RS (6,2)	6	2	8	1.33×	2 shards (25%)
RS (8,4)	8	4	12	1.50×	4 shards (33.3%)

means of detecting differences between the stored and retrieved data, the GCM authentication tag provides cryptographic authentication during decryption [15]. Combining these mechanisms strengthens the framework by addressing data integrity and confidentiality through separate but complementary security operations [4], [15].

The use of SHA-256 also supports reliable evaluation of the reconstruction process. By comparing the computed digest with the stored reference value, the system can verify whether the reconstructed ciphertext matches the data originally submitted for storage.

3.5 Reed-Solomon Erasure Coding Configurations

Reed-Solomon erasure coding is employed in the proposed framework to improve storage efficiency while maintaining

data availability in a distributed environment. Unlike replication, which stores multiple complete copies of the same data, erasure coding divides the input into data and parity shards. The original data can be reconstructed from a sufficient number of available shards, even when some shards become unavailable [3], [9]. The technique is based on error-correcting codes developed by Reed and Solomon and is widely applicable to reliable data storage systems [14].

In this study, three Reed-Solomon configurations, namely (4,2), (6,2), and (8,4), are considered for comparative evaluation. Each configuration is represented by the number of data shards (k) and parity shards (m). The total number of shards is determined by the sum of these two values. Under the assumption that shards are distributed across separate storage nodes, the parity count determines the maximum number of shard failures that can be tolerated while retaining the ability to reconstruct the original data [12].

The storage factor of each configuration is calculated as the ratio of the total number of shards to the number of data shards. A lower storage factor indicates reduced storage requirements relative to the original data size, whereas a higher parity count can provide greater tolerance to shard failures. Therefore, selecting an appropriate configuration involves balancing storage consumption, fault tolerance, and processing requirements.

Table 1. Reed-Solomon Configurations and Their Theoretical Characteristics

As shown in Table 1, RS (6,2) has the lowest storage factor among the evaluated configurations, requiring approximately 1.33 times the original data size. RS (4,2) and RS (8,4) have a storage factor of 1.50 times the original data size. In terms of theoretical fault tolerance, RS (8,4) can tolerate the loss of up to four shards, while RS (4,2) and RS (6,2) can tolerate the loss of up to two shards, provided that the remaining shards are sufficient for reconstruction. These differences form the basis for evaluating the trade-offs between storage overhead, fault tolerance, and reconstruction performance in the subsequent experimental analysis.

3.6 Data Distribution and Reconstruction Process

After encryption and integrity verification, the processed data is divided into data and parity shards using the selected Reed-Solomon configuration. These shards are distributed across the MinIO S3-compatible storage cluster to support data availability and fault tolerance. The distribution process enables the system to store different portions of the encoded data across multiple storage nodes rather than maintaining complete replicas of the original file [9], [12].

During data storage, the system retains the necessary metadata, including the file identifier, erasure-coding configuration, shard information, and reference hash value. This metadata supports the identification and reconstruction of the stored object during subsequent retrieval operations. The use of erasure coding allows the system to recover the original data when the number of unavailable shards remains

within the fault-tolerance limit of the selected configuration [3], [12].

When a file is requested, the system retrieves the available shards from the storage cluster and applies Reed-Solomon decoding to reconstruct the encrypted data. If sufficient shards are available, the reconstruction process can recover the original ciphertext without requiring every shard to be present [14]. The reconstructed ciphertext is then verified against the stored SHA-256 reference digest to check its integrity. Following successful verification, AES-256-GCM decryption and authentication-tag validation are performed to recover the original file [15].

The reconstruction process is evaluated by comparing the recovered file with the original input. Successful recovery is established when the reconstructed output matches the original data byte-for-byte. This approach provides a practical basis for assessing the reliability of the proposed framework under the tested shard-failure conditions.

IV. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

4.1 Experimental Environment

The proposed secure distributed storage framework was evaluated through controlled experiments to examine the performance of different Reed-Solomon configurations. The experimental setup integrates client-side AES-256-GCM encryption, SHA-256 integrity verification, Reed-Solomon erasure coding, and MinIO-based S3-compatible object storage. These components were combined to assess the interaction between data security, storage overhead, and reconstruction performance. The use of erasure coding in distributed storage environments provides a practical approach to balancing data availability and storage efficiency [3], [9].

Three erasure-coding configurations, namely RS (4,2), RS (6,2), and RS (8,4), were selected for evaluation. Each configuration was tested using the same data-processing workflow to maintain consistency across the experiments. The evaluation considered file sizes of 500 MB and 1 GB, with the 500 MB input used for reconstruction verification and the 1 GB input used for the reported processing-time comparison.

The experimental framework recorded the time required to process the input data and assessed the ability to reconstruct the encrypted file following shard removal. Data integrity was verified by comparing the SHA-256 digest of the reconstructed ciphertext with the stored reference digest. The recovered output was also checked against the original input to establish whether reconstruction preserved the original data.

The experiments were conducted on an HP Pavilion Gaming Laptop (15-dk2xxx) equipped with an 11th Generation Intel Core i5-11300H processor operating at 3.10 GHz, featuring four physical cores and eight logical processors. The system was configured with 8 GB of installed RAM and Microsoft

Windows 11 Home Single Language (64-bit). This hardware environment was used to execute the proposed framework and evaluate the performance of the selected Reed-Solomon configurations.

4.2 Performance Evaluation Metrics

The performance of the proposed framework was evaluated using metrics that represent storage efficiency, processing performance, fault tolerance, and data integrity. These metrics provide a basis for comparing the selected Reed-Solomon configurations under a consistent experimental workflow.

Storage Factor: The storage factor represents the total storage required relative to the original data size, including both data and parity shards. It is calculated as the ratio of total shards to data shards. A lower storage factor indicates greater storage efficiency because less additional storage is required for a given amount of original data [3], [12].

Processing Time: Processing time measures the duration required to complete the client-side data-processing operations, including AES-256-GCM encryption, SHA-256 hash computation, and Reed-Solomon erasure coding. The measurement begins when the input file enters the processing workflow and ends after the encoded shards are generated. This metric is used to compare the computational performance of different erasure-coding configurations under the same experimental conditions.

Fault Tolerance: Fault tolerance refers to the maximum number of unavailable shards that a configuration can withstand while still allowing reconstruction of the original data. For a Reed-Solomon configuration with k data shards and m parity shards, reconstruction is theoretically possible when at least k shards remain available [9], [14].

Data Integrity: Data integrity is evaluated by comparing the SHA-256 digest of the reconstructed ciphertext with the reference digest generated before storage. A matching digest indicates that the reconstructed ciphertext is consistent with the originally stored data [4].

Reconstruction Accuracy: Reconstruction accuracy is assessed by comparing the recovered file with the original input at the byte level. This check determines whether the reconstruction process preserves the original file content without introducing differences. The authentication mechanism of AES-GCM provides an additional verification step during decryption [15].

4.3 Experimental Procedure

The experiments followed a consistent sequence to ensure that the selected Reed-Solomon configurations could be compared under equivalent processing conditions. Initially, an input file of the selected size was provided to the client-side application. The file was encrypted using AES-256-GCM, after which its SHA-256 digest was generated and retained as a reference for integrity verification [4], [15].

The encrypted data was then divided into data and parity shards according to the selected Reed-Solomon configuration and distributed across the MinIO storage cluster. After the storage operation, selected shards were made unavailable to examine the reconstruction capability of the framework. The remaining shards were retrieved and processed using Reed-Solomon decoding to recover the encrypted data [12], [14].

Following reconstruction, the SHA-256 digest was recalculated and compared with the stored reference value. The reconstructed ciphertext was then subjected to AES-256-GCM decryption and authentication-tag verification. Finally, the recovered file was compared with the original input to determine whether the data had been restored correctly [4], [15].

The experimental procedure was applied across the selected Reed-Solomon configurations and input sizes. The 500 MB input size was also used for reconstruction verification, while the processing-time comparison presented in Table 2 focuses on the 1 GB measurements. The number of trials varied depending on the configuration and input size. The recorded measurements were used to compare processing time, storage factor, and reconstruction behaviour. The resulting observations are presented and discussed in the following section.

V. RESULTS AND DISCUSSION

5.1 Processing Time Analysis

The processing time of the proposed secure distributed storage framework was evaluated using three Reed-Solomon configurations: RS (4,2), RS (6,2), and RS (8,4). The evaluation considered the time required to process a 1 GB input file through the client-side encryption, integrity verification, and erasure coding workflow. All configurations were evaluated under the same controlled experimental setup to facilitate a comparative analysis.

The measured processing times are presented in Table 2.

Table 2. Processing Time Comparison for a 1 GB Input File

Reed-Solomon Configuration	Processing Time (seconds)
RS (4,2)	6.5965
RS (6,2)	5.9419
RS (8,4)	9.3975

The experimental results indicate that RS (6,2) recorded the lowest processing time of 5.9419 seconds, followed by RS (4,2) at 6.5965 seconds. The RS (8,4) configuration required 9.3975 seconds, representing the highest processing time among the evaluated configurations.

The observed differences can be associated with the computational overhead introduced by the erasure coding process. Reed-Solomon coding involves encoding data into data and parity fragments, and the computational requirements depend on the selected coding parameters and implementation [3], [9], [14].

Configurations involving a larger number of shards may introduce additional encoding and data distribution overhead, which can influence overall processing time [12].

Although RS (8,4) provides a higher theoretical shard-failure tolerance, its measured processing time was greater than that of the other two configurations. In comparison, RS (6,2) demonstrated the lowest processing time in the conducted 1 GB experiment. These observations highlight the importance of considering processing overhead alongside fault tolerance when selecting an erasure coding configuration for distributed storage systems [3], [12].

It should be noted that the reported 1 GB measurements represent one trial per configuration. Therefore, the results provide an initial comparative observation rather than a statistically conclusive performance assessment. Further repeated experiments under identical conditions would be required to establish the consistency of the observed differences.

5.2 Storage Factor Comparison

Storage efficiency is an important parameter in distributed storage systems because it determines the additional storage capacity required to provide data redundancy and fault tolerance. Unlike replication-based approaches, erasure coding divides the original data into data and parity fragments, enabling recovery from shard failures with comparatively controlled storage requirements [3], [9].

The storage factor of the three evaluated Reed-Solomon configurations was calculated as the ratio of the total number of data and parity shards to the number of data shards. The results are presented in Table 3.

Table 3. Storage Factor Comparison of Reed-Solomon Configurations

Configuration	Data Shards	Parity Shards	Total Shards	Storage Factor
RS (4,2)	4	2	6	1.50×
RS (6,2)	6	2	8	1.33×
RS (8,4)	8	4	12	1.50×

The results show that RS (6,2) requires a storage factor of approximately 1.33×, indicating that the total encoded storage is approximately 33.3% greater than the original data size. In

comparison, RS (4,2) and RS (8,4) require a storage factor of 1.50×, corresponding to a 50% storage overhead.

The difference in storage requirements is directly related to the proportion of parity shards in each configuration. A higher proportion of parity shards increases redundancy and can improve the ability to recover data from shard failures, but it also increases storage requirements [3], [9], [12]. Therefore, storage efficiency and fault tolerance represent important considerations when selecting an erasure coding configuration for distributed object storage.

Based on the calculated storage factors, RS (6,2) demonstrates lower storage overhead among the evaluated configurations. However, storage efficiency should not be considered independently of processing time and fault tolerance, as these parameters collectively influence the suitability of a configuration for a particular deployment scenario.

5.3 Fault Tolerance and Reconstruction Analysis

Fault tolerance is a critical characteristic of distributed storage systems, as it determines the ability to recover data when one or more storage fragments become unavailable. Reed-Solomon erasure coding provides fault tolerance by generating parity fragments that can be used to reconstruct the original data in the presence of shard failures [3], [9], [14].

The theoretical fault tolerance of the evaluated configurations is determined by the number of parity shards. The corresponding results are presented in Table 4.

Table 4. Theoretical Fault Tolerance of Reed-Solomon Configurations

Configuration	Total Shards	Parity Shards	Maximum Tolerable Shard Failures	Failure Tolerance
RS (4,2)	6	2	2	33.3%
RS (6,2)	8	2	2	25.0%
RS (8,4)	12	4	4	33.3%

As shown in Table 4, RS (4,2) and RS (6,2) can theoretically tolerate the failure of up to two shards, whereas RS (8,4) can tolerate up to four shard failures. These capabilities are determined by the parity structure of each configuration and are consistent with the fundamental properties of Reed-Solomon erasure coding [3], [12], [14].

To evaluate reconstruction capability, shard removal experiments were conducted to simulate storage node or fragment unavailability. The reconstruction process was used to recover the original encrypted data from the remaining available fragments. The recovered data was then compared with the original data to verify successful reconstruction.

The experimental results confirmed byte-for-byte successful reconstruction in the conducted trials. This demonstrates that the evaluated configurations were capable of recovering the original data under the tested shard-removal conditions. However, the experimental validation involved the removal of two shards; therefore, the theoretical four-shard tolerance of RS (8,4) should not be interpreted as experimentally verified in this study.

The findings indicate that Reed-Solomon erasure coding can provide reliable data recovery while allowing different levels of redundancy and storage overhead. The selection of a configuration should therefore consider the required fault tolerance, storage capacity, and processing overhead together [3], [9], [12].

5.4 Data Integrity Verification

Data integrity is an essential requirement for secure distributed storage systems, as data must remain accurate and unaltered during encryption, distribution, storage, and reconstruction. In the proposed framework, SHA-256 hashing and AES-256-GCM authentication are employed to support data integrity verification [4], [15].

During the upload process, a SHA-256 digest is generated for the encrypted data before erasure coding. The resulting digest is retained as a reference value for subsequent verification. During retrieval, the encrypted data is reconstructed from the available shards, and a new SHA-256 digest is calculated. The reconstructed ciphertext is considered consistent with the stored reference when both digest values match.

In addition to hash-based verification, AES-256-GCM provides authenticated encryption by combining confidentiality with authentication. The authentication tag is verified during decryption to detect unauthorized modifications to the ciphertext [13], [15].

The experimental reconstruction results demonstrated byte-for-byte consistency in the conducted trials. This observation supports the correctness of the reconstruction process and the integrity verification approach implemented in the proposed framework.

The combined use of SHA-256 and AES-256-GCM provides complementary integrity checks. SHA-256 enables comparison of the reconstructed encrypted data against its reference digest, while GCM authentication verifies the integrity and authenticity of the encrypted content during decryption. These mechanisms contribute to the reliability of the proposed secure distributed storage framework [4], [13], [15].

5.5 Overall Discussion

The experimental evaluation highlights the trade-offs among processing time, storage efficiency, fault tolerance, and data integrity in the proposed secure distributed S3 storage framework. The three evaluated Reed-Solomon configurations exhibited different performance characteristics, demonstrating that configuration selection

involves balancing computational efficiency and data redundancy.

Among the evaluated configurations, RS (6,2) recorded the lowest processing time of 5.9419 seconds for the 1 GB input file and required a storage factor of approximately 1.33×. In comparison, RS (4,2) required 6.5965 seconds with a storage factor of 1.50×, while RS (8,4) required 9.3975 seconds with the same storage factor of 1.50×. These results illustrate the relationship between coding configuration, processing overhead, and storage efficiency.

From a fault tolerance perspective, RS (8,4) provides a theoretical tolerance of four shard failures, compared with two shard failures for RS (4,2) and RS (6,2). However, this increased theoretical recovery capability is associated with a higher measured processing time in the conducted experiment. Such trade-offs are consistent with the broader considerations involved in selecting erasure coding schemes for distributed storage environments [3], [9], [12].

The reconstruction experiments confirmed successful byte-for-byte reconstruction in the conducted trials, supporting the reliability of the implemented recovery process. Furthermore, the combination of SHA-256 digest comparison and AES-256-GCM authentication provides complementary mechanisms for verifying data integrity and authenticity [4], [13], [15].

Overall, the findings demonstrate that the evaluated Reed-Solomon configurations offer different combinations of processing performance, storage efficiency, and theoretical fault tolerance within the proposed secure storage framework. The results provide a basis for selecting configurations according to application-specific requirements. However, the limited number of 1 GB trials and the absence of experimental validation for the maximum theoretical shard-failure limits should be considered when interpreting the findings. Further experiments involving repeated measurements, larger datasets, and additional failure scenarios would help establish more comprehensive performance characteristics.

VI. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This study presented a comparative analysis of three Reed-Solomon configurations, namely RS (4,2), RS (6,2), and RS (8,4), within a secure distributed S3 storage framework integrating AES-256-GCM encryption, SHA-256 integrity verification, and MinIO object storage. The primary objective was to evaluate the trade-offs among processing time, storage efficiency, fault tolerance, and data integrity.

The experimental results demonstrated that RS (6,2) achieved the lowest processing time of 5.9419 seconds for a 1 GB input file, with a storage factor of approximately 1.33×. RS (4,2) recorded a processing time of 6.5965 seconds and a storage factor of 1.50×, whereas RS (8,4) required 9.3975 seconds with the same storage factor of 1.50×. These findings highlight the impact of erasure coding configuration on processing performance and storage utilization.

The theoretical fault tolerance analysis indicated that RS (4,2) and RS (6,2) can tolerate up to two shard failures, while RS (8,4) can theoretically tolerate up to four shard failures. The conducted reconstruction experiments confirmed byte-for-byte consistency of the recovered data under the tested conditions. Furthermore, the integration of SHA-256 digest comparison and AES-256-GCM authentication provided complementary mechanisms for verifying data integrity and authenticity.

Overall, the study demonstrates the feasibility of integrating encryption, integrity verification, and Reed-Solomon erasure coding within a distributed S3-compatible storage framework. The findings emphasize that configuration selection should consider processing efficiency, storage requirements, and fault tolerance according to the intended application. The results also provide a foundation for further investigation of secure and reliable distributed storage architectures.

6.2 Future Scope

Future work can extend the present study in several directions to improve the performance, scalability, and reliability of the proposed framework.

1. Large-Scale Performance Evaluation:

Experiments can be extended to larger datasets and multiple repeated trials to obtain more consistent performance measurements and improve the statistical reliability of the results.

2. Expanded Fault-Tolerance Testing:

Additional shard-removal experiments can be conducted to validate the maximum theoretical failure tolerance of each Reed-Solomon configuration, including four-shard failure scenarios for RS (8,4).

3. Distributed Cluster Evaluation: The framework can be evaluated across multiple physical or virtual storage nodes to investigate the effects of network latency, node failures, and concurrent operations on overall performance.

4. Dynamic Configuration Selection: An adaptive mechanism can be developed to select Reed-Solomon configurations based on workload characteristics, available storage capacity, and fault-tolerance requirements.

5. Performance Optimization: Further optimization of encryption, hashing, and erasure coding operations can be explored to reduce processing overhead while maintaining data confidentiality and integrity.

6. Scalability and Concurrent Access: Future experiments can investigate simultaneous uploads, downloads, and reconstruction requests to evaluate the framework under more demanding storage workloads.

These extensions can provide a more comprehensive understanding of the performance and reliability of secure distributed storage systems and support their deployment in practical cloud storage environments.

REFERENCES

- [1] Wang and Zhao, "Client-Side Encryption Standards in S3-Compatible Object Storage Clusters," *Journal of Cloud Computing and Cybersecurity*, vol. 9, no. 4, pp. 215–230, 2020.
- [2] Li and Chen, "The Cost of Redundancy: Analyzing the Overhead of 3-Way Replication in Distributed Systems," *BDE*, pp. 45–52, 2021.
- [3] Zhang et al., "Fault Tolerance and Reliability Analysis of Erasure Coding vs. Replication in Cloud Storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 2, pp. 310–325, 2019.
- [4] Smith and Williams, "Detecting Silent Data Corruption: The Role of SHA-256 and HMAC in Cold Storage," *Journal of Data Integrity*, vol. 14, no. 1, pp. 88–102, 2022.
- [5] Agrawal et al., "Performance Analysis of MinIO Object Storage in Cloud-Native Environments," *IEEE Cloud Computing*, vol. 9, no. 3, pp. 44–53, 2022.
- [6] Garcia and Martinez, "Hybrid Security Models: Integrating AES-256 and Sharding for Multi-Cloud Environments," in *Proceedings of the IEEE Security & Privacy Symposium*, pp. 567–580, 2023.
- [7] Kumar and Singh, "Optimizing Galois Field Arithmetic for Reed-Solomon Encoding in Real-Time Applications," *Journal of Mathematical Computing*, vol. 11, no. 2, pp. 401–415, 2020.
- [8] Rani and Garg, "Performance Overhead of Encryption in Cloud Storage Systems," *International Journal of Computer Applications*, vol. 183, no. 6, pp. 1–7, 2021.
- [9] H. Huang et al., "Erasure Coding in Windows Azure Storage," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, pp. 15–26, 2012.
- [10] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," in *Financial Cryptography and Data Security*, Springer, pp. 136–149, 2010.
- [11] Lewis, "Dynamic Sharding Strategies for Heterogeneous Storage Clusters," *ACM Transactions on Storage*, vol. 18, no. 3, pp. 20–35, 2022.
- [12] MinIO, "Erasure Code Calculator and Deployment Guide for High Availability," MinIO Documentation, 2023. [Online]. Available: <https://min.io/docs/minio/linux/operations/concepts/erasure-coding.html>
- [13] National Institute of Standards and Technology (NIST), *Advanced Encryption Standard (AES)*, FIPS PUB 197, 2001.

Updated May 2023. [Online]. Available:
<https://csrc.nist.gov/pubs/fips/197/final>

[14] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[15] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D, National Institute of Standards and Technology, 2007. doi: 10.6028/NIST.SP.800-38D.

[16] T. Nagwanshi, "Performance Optimization of Distributed S3 Clusters Through Erasure Coding and Advanced AES-256 Encryption," *International Research Journal of Modernization in Engineering Technology and Science (IRJMETS)*, vol. 8, no. 4, pp. 12415–12422, 2026. doi: 10.56726/IRJMETS96725.