Journal of Theoretical and Applied Information Technology
31st August 2024. Vol.102. No. 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCING CLOUD DATA SECURITY THROUGH LONG-TERM SECRET SHARING SCHEMES

SARA IBN EL AHRACHE[1], HASSAN BADIR[2]
[1,2] IDS Team, School of Applied Sciences of Tangier, Abdelmalek Essaadi University, Morocco
E-mail: [1]selahrache@uae.ac.ma, [2]hbadir@uae.ac.ma

ABSTRACT

Cloud computing has experienced significant growth in recent years, becoming a cornerstone of modern IT infrastructure. Promising "infinite scalability and unlimited resources," cloud service providers offer on-demand access that often obscures the underlying computing infrastructure. The inherent complexity of virtualized, multi-tenant cloud environments surpasses that of traditional data centers, complicating service management, particularly in terms of security. Despite these challenges, the appealing features of cloud computing have led many organizations to adopt cloud storage services for their critical data. Users can store data remotely in the cloud and access it via thin clients when needed. However, data security remains a paramount concern due to the internet-based nature of cloud services, which limits user control over stored data. This paper proposes an innovative approach to enhance data security in cloud environments through a Long-Term Secret Sharing Scheme (SSS-LT). Secret sharing schemes partition and distribute data across multiple cloud service providers, thereby increasing data privacy and availability. Our proposed SSS-LT addresses a key limitation of existing secret sharing methods: the degradation of computational performance with large data sets. We conduct a theoretical analysis of the security and complexity factors influencing our approach and validate its efficacy through experimental evaluation, demonstrating its superiority over existing methods.

Keywords: *Cloud Computing, Data Security, Secret Sharing, Map Reduce*

1. INTRODUCTION

Cloud computing is heralded as a transformative paradigm poised to revolutionize the consumption of computing resources. Despite its numerous benefits, cloud computing has brought forth specific security issues that are now a primary focus of research in this field. One of the most significant challenges in adopting cloud services is convincing users to trust the security of these services enough to store their sensitive data. Although cloud service providers often boast sophisticated encryption mechanisms, traditional cloud systems cannot guarantee data security if cloud servers are compromised. This inherent insecurity is exacerbated in cloud environments due to their unique characteristics, including seamless scalability, shared resources, multi-tenancy, ubiquitous access, and high availability on demand.

A substantial body of research has identified various security and privacy issues specific to cloud computing. These issues are generally classified into two categories:

Amplified Cloud Security Issues -- These are existing problems in traditional distributed computing environments that are magnified by the characteristics of cloud computing.

Specific Cloud Security Issues -- These are new security challenges that arise due to the unique features of cloud computing.

A comprehensive review of the literature, best practices, and standard recommendations on cloud computing security reveals several critical security issues:

- Misuse of Administrator Rights / Malicious Collaborators: The threat posed by the misuse of administrator rights is amplified in cloud computing. Virtual machines (VMs) are often provided as root servers, giving cloud providers access to VMs via the hypervisor, which can be misused by malicious insiders.
- Lack of Transparency of Security Measures: Cloud computing often lacks transparency regarding the security

measures and processes applied by cloud providers, forcing customers to rely on vendor assurances without verifiable evidence.

- Lack of Transparency in Security Incidents: In cloud environments, both the client and the vendor must collaborate to address security incidents. However, there is currently no standardized procedure for such collaboration.
- Shared Technology Issues: The shared use of physical resources and the potential for compromised VMs to affect others are significant security risks due to inadequate isolation in virtualization.
- Data Lifecycle Management: The challenge of securely managing data at the end of a contract is heightened in cloud computing due to the shared use of resources.
- Monitoring Service Level Agreements (SLAs): Monitoring SLAs in a multi-tenant cloud environment requires specialized tools for hypervisor and virtualized network monitoring, which are currently insufficient.

Additionally, cloud-specific security issues include:

- Imprecise Location of Data: Customers often do not know the precise location of their data within a cloud provider's infrastructure, which complicates data sovereignty and compliance concerns.
- Abuse and Harmful Use of Cloud Resources: The rapid provisioning of VMs in the cloud can be exploited for malicious activities, such as launching Distributed Denial of Service (DDoS) attacks.
- Lack of Supervision: Cloud providers are responsible for detecting and mitigating security incidents, but there is a lack of automated reporting systems to inform clients about such incidents.
- Non-Secure Application Programming Interfaces (APIs): The security of cloud services depends on the security of vendor-specific APIs, which must be robust to prevent unauthorized access and malicious attacks.
- Lack of Cloud Scalability Monitoring: Cloud users often rely on the scalability of cloud infrastructure to handle peak demands. However, adequate monitoring tools to manage this scalability are often lacking.
- Absence of Interoperability among Cloud Service Providers: The lack of compatibility between different cloud providers' services increases the risk of vendor lock-in and complicates the migration of resources between providers.
- Increased Complexity due to Cloud Characteristics: The complexity of cloud environments necessitates a re-evaluation of traditional best practices in security and business continuity, as they may not be sufficient to mitigate security incidents.

With the rapid adoption of cloud computing, organizations are migrating sensitive and critical data to cloud environments. This transition has heightened concerns about data security and privacy, as cloud environments expose data to various risks, including unauthorized access, breaches, and loss. The increasing reliance on cloud services necessitates more robust security solutions to protect sensitive information from these evolving threats.

Despite the development of various security mechanisms, traditional methods are often insufficient for addressing the unique challenges posed by cloud computing. For instance, traditional encryption techniques may not effectively handle the complex data sharing and access control requirements in multi-tenant cloud environments. The lack of transparency in security measures and incident management, coupled with issues such as shared resources and inadequate isolation, highlights the need for innovative solutions that offer improved security and reliability.

Current secret sharing schemes, while effective in certain contexts, have limitations in terms of computational performance and scalability when dealing with large data sets in cloud environments. Traditional schemes may struggle with the performance overhead of processing and managing large volumes of data, which can hinder their effectiveness in real-world applications. This creates a need for new approaches that can handle large data sets more efficiently while maintaining strong security guarantees.

As cloud environments continue to evolve, there is a growing demand for solutions that enhance both data privacy and availability. The proposed Long-Term Secret Sharing Scheme (SSS-LT) aims to

address these needs by fragmenting and distributing data across multiple cloud service providers. This approach not only improves data privacy but also ensures higher availability, reducing the risk of data loss and unauthorized access.

The proposed SSS-LT offers a novel approach that fills existing gaps in the literature and addresses the limitations of current methods. By integrating secret sharing with cloud-specific requirements and demonstrating its superiority through theoretical analysis and experimental validation, this research contributes a significant advancement to the field of cloud security. It provides a robust solution to pressing issues in data security, offering a new perspective on managing and protecting sensitive information in cloud environments.

In summary, this research is required due to the growing reliance on cloud computing, existing gaps in current security measures, and the need for improved privacy and availability. The proposed Long-Term Secret Sharing Scheme (SSS-LT) addresses these challenges, offering a significant contribution to enhancing cloud data security.

2. RELATED WORK

Significant research efforts have been directed towards enhancing the security of cloud computing environments, particularly through the use of secret sharing schemes. These schemes provide a robust method for securing data by dividing it into multiple parts and distributing these parts across different locations or servers, thereby reducing the risk of data breaches.

AlZain et al. (2012) introduced a model leveraging Multi-Clouds Databases (MCDB) and secret sharing algorithms to secure cloud services. Their approach employs Triple Modulus Redundancy (TMR) and the sequential method to enhance reliability. The MCDB model utilizes the Database-as-a-Service (DaaS) offerings from multiple cloud providers, using secret sharing algorithms like Shamir's to ensure data security and prevent sensitive data leakage. The TMR method employs three identical modules or machines performing the same task in parallel, with the output determined by majority voting. This method, combined with secret sharing algorithms, significantly enhances data security and reliability by mitigating the risk of data breaches .

Alsolami and Boult (2014) proposed an approach combining Shamir's secret sharing with hashing and signature of shares to ensure data integrity and fault tolerance. They implemented multi-threading in the remote downloading and uploading of shares to improve system performance. This method ensures that the data remains secure and intact even in the presence of faults or attacks .

Lee et al. (2014) concluded that multi-cloud environments offer superior security, integrity, and availability compared to single-cloud systems. They proposed integrating a homomorphic encryption system into the DepSky approach, which already combines Byzantine Fault Tolerance (BFT) and erasure code cryptography. This integration aims to enhance the security of secret sharing algorithms, particularly when handling sensitive data .

Fabian et al. (2015) presented a comprehensive security framework incorporating several measures, such as authentication to prevent unauthorized access, network security through TLS to encrypt communications, federated identity management, and access control to medical records based on Role-Based Access Control (RBAC). They utilized Shamir's secret sharing and Rabin's Information Dispersal Algorithm to secure data replication across multiple cloud providers.

Ke et al. proposed a dual-threshold secret sharing scheme (T, m) - (k, n), where the two thresholds are used for secret and mask sharing. Their approach involves two variants: Computational Security and Information Theoretic Security. The former uses a one-way function to generate a key and share the secret, while the latter ensures security even against opponents with unlimited computational power. This method involves generating a random number, XORing it with the secret, and sharing the result across multiple servers.

Recent advancements have also been made in cloud security through the application of block chain technology. Liu et al. (2018) proposed a blockchain-based approach to enhance the security and privacy of cloud data storage. Their method leverages the decentralized nature of blockchain to distribute and store data securely, ensuring integrity and preventing unauthorized access. Similarly, Zhang et al. (2019) introduced a blockchain-based data sharing framework that utilizes smart contracts to manage data access permissions, enhancing transparency and security in cloud environments .

Yang et al. (2020) developed a secure and efficient cloud data storage scheme using a verifiable secret sharing mechanism. Their approach ensures data security and integrity through the use of

cryptographic techniques and provides verifiability to detect any unauthorized modifications

Chen et al. (2021) proposed a hybrid cloud security model that combines secret sharing and homomorphic encryption to secure sensitive data in cloud environments. Their method ensures data confidentiality and supports secure data computation in the cloud without revealing the data to cloud service providers .

In summary, various approaches and models have been proposed to address the security challenges in cloud computing. These include leveraging multi-cloud environments, combining secret sharing algorithms with hashing and multi-threading, integrating homomorphic encryption, and utilizing block chain technology. Each method offers unique advantages in enhancing data security, integrity, and availability in the cloud.

3. PROBLEM STATEMENT

The rapid adoption of cloud computing has led to an unprecedented volume of data being stored and processed in cloud environments. This shift towards cloud storage has introduced significant concerns regarding data security and privacy. While cloud computing offers numerous advantages, including scalability, cost-effectiveness, and accessibility, it also presents unique security challenges that are not adequately addressed by traditional methods.

Data breaches and information leakage from cloud systems have emerged as major concerns, driven by vulnerabilities in cloud infrastructure and the complex nature of managing sensitive data across multiple tenants. Despite advancements in cloud security, existing solutions often fall short in addressing the comprehensive security needs of cloud environments. Traditional encryption and access control mechanisms, while effective in some contexts, do not fully account for the dynamic and multi-tenant nature of cloud computing, where data is distributed across multiple servers and providers.

Recent literature highlights several key challenges:

- Data Privacy and Security: Cloud environments involve storing data across multiple servers and potentially across multiple providers, raising concerns about data privacy and the security of data during storage and transmission. Traditional encryption techniques may not adequately protect data from sophisticated threats and vulnerabilities inherent in cloud computing (Chen et al., 2021; Yang et al., 2020).
- Efficiency and Performance: Secret sharing schemes, while providing robust security guarantees, often suffer from performance overhead, especially when handling large data sets. This performance issue is exacerbated in cloud environments where efficiency is critical (Ke et al., n.d.; Alsolami and Boult, 2014).
- Scalability: The scalability of cloud services introduces additional complexity in data management and security. Current solutions may struggle to scale effectively while maintaining high levels of security and performance (AlZain et al., 2012; Fabian et al., 2015).
- Integration with Cloud-Specific Architectures: Many existing secret sharing schemes are not optimized for integration with cloud-specific architectures, such as distributed storage systems and virtualized environments. This gap creates a need for novel approaches that align with the unique characteristics of cloud computing (Lee et al., 2014; Liu et al., 2018).

The proposed research aims to address these challenges by integrating secret sharing schemes within the Hadoop Map Reduce architecture. This approach leverages the inherent benefits of Map Reduce—such as parallel processing and cost-effectiveness—while enhancing data security through advanced secret sharing techniques. The Long-Term Secret Sharing Scheme (SSS-LT) proposed in this research is designed to improve both the efficiency and security of data processing in cloud environments. By fragmenting and distributing data across multiple cloud service providers, the SSS-LT approach aims to address the limitations of existing methods and provide a more robust solution to the problem of data security in cloud computing.

This research is crucial because it addresses current gaps in cloud security solutions, particularly in terms of performance and scalability. The integration of secret sharing with cloud-specific architectures represents a significant advancement in securing sensitive data and ensuring its availability, contributing valuable insights and practical solutions to the field of cloud security.

4. PRPOSED SOLUTION

4.1 Preliminaries

4.1.1 Shamir Secret Sharing

Shamir and Blakley introduced straightforward yet robust sharing schemes that allow a threshold k of n participants, where k <= n, to reconstruct a secret. Shamir's solution utilized polynomial curves, enabling the secret to be reconstructed through interpolation when at least k participants provided their shares.

Definition: In a secret sharing threshold scheme (k, n), a secret s∈F can be divided into parts S_1, S_2 ,…., S_n such that:

- Any set of k or more shares can reconstruct the secret.
- Any set of fewer than k shares reveals no information about the secret.

Shamir's Secret Sharing Scheme: Given n participants, P = {p_1,p_2,….,p_n }, polynomial interpolation can be used to create a secret sharing threshold scheme (k, n). To successfully reconstruct the secret, a subset A ⊆ P with |A|≥k is required.

Shares Creation: The dealer D selects a secret s∈F and constructs a random polynomial f(x) of degree *k* – 1 : $f(x) = s + r_1x + r_2x^2 + r_{(k-1)}x^{(k-1)}$ modF

Subject to the following conditions:

- The field F is a Galois Field GF(q) where F>n and q is a prime power.
- The secret s∈F.
- The threshold is k.
- The coefficients r_1 ….. r_{k-1} are chosen independently and randomly from the interval [0,F).
- Each part Si of the secret is created by evaluating the function f(x): $S_1 = f(1)$, $S_2 = f(2)$, ….., $S_n = f(n)$

Secret Reconstruction: The secret can be reconstructed using polynomial interpolation. At least k participants (the degree of the polynomial plus one) must contribute their shares to reconstruct the polynomial. Each participant provides a consistent pair consisting of the value of x and the result of the polynomial function at x (i.e., each participant has a pair (x, $q(x) = S_x$). Since no two participants share the same x value, the pairs form a Lagrange basis. The interpolation polynomial in Lagrange form is defined as: $L(z) == (i = 1)^k q(x_i).(j = 1)(k, ji)(z - x_j)/(x_i - x_j)$ modF.

4.1.2 Map Reduce Paradigm

Map Reduce is a programming paradigm designed for the distributed parallel processing of large datasets. It transforms these datasets into tuples and then combines and reduces them into smaller tuples. Fundamentally, Map Reduce was created to handle large amounts of data (terabytes or more) by utilizing parallel distributed computing to convert significant data into smaller, more manageable chunks. The basic idea is to address data distribution over a network of computers to maximize the use of available memory, processors, and storage. This allows programmers to focus on the unique aspects of their processing tasks.

In Map Reduce, the developer only needs to define two functions—Map and Reduce—while the implementation handles the rest. Although there are many additional features and settings to fine-tune the model, the core functionality remains unchanged. A Map Reduce computation follows these steps:

1. Input Reading: The input is read from disk and converted into key-value pairs.
2. Mapping: The Map function processes each pair separately and outputs the results as several key-value pairs.
3. Reducing: For each distinct key, the Reduce function processes all key-value pairs associated with that key and returns any number of key-value pairs.
4. Output Writing: Once all input pairs are processed, the Reduce function's output is written to the disk as key-value pairs.

Parallel distributed processing allows large volumes of data to be processed quickly by distributing tasks across groups of servers. In Map Reduce, tuples refer to key-value pairs by which data is grouped, sorted, and processed. Map Reduce jobs execute sequences of map and reduce processes on a distributed set of servers.

During the map phase, data is converted into key-value pairs, transformed, and filtered, then assigned to nodes for processing. In the reduce phase, the data is condensed into smaller datasets. The data is transformed into a standard keyframe format where the key serves as a record identifier and the value is the corresponding data. The process involves two main steps:

1. Mapping the Data: Incoming data is delegated into key-value pairs and divided into fragments assigned to Map tasks. Each cluster (a group of connected nodes performing a shared task) receives several Map tasks, distributed among its nodes. Processing key-value pairs generates intermediate key-value pairs, which are sorted by their key values and divided into new fragments.
2. Reducing the Data: Each reduction task processes a fragment and produces an output, also in key-value pairs. Reduction tasks are distributed among the cluster nodes. Once the task is completed, the final output is written to a file system. A diagram of the Map Reduce architecture is presented in Figure 1.

Journal of Theoretical and Applied Information Technology
31st August 2024. Vol.102. No. 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

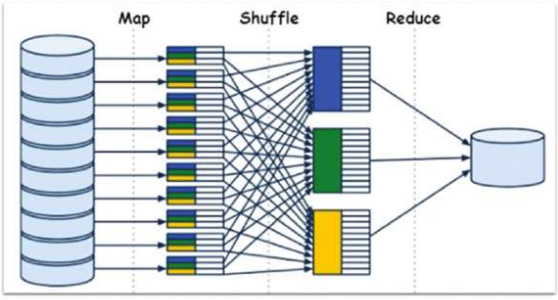

Figure 1: Map Reduce Architecture

It is important to note that the MapReduce model specifies a general structure for how data is transmitted and processed but does not detail the specific steps of calculation with the data—these are defined by the user for the four main steps.

4.2 Proposed Scheme

4.2.1 Proposed System Principle

Secure data storage in the Cloud is a challenging task, especially with the increasing data sizes that traditional algorithms struggle to handle. In this section, we propose a scalable design and implementation of the Secret Sharing scheme using Map Reduce to manage large datasets. The goal is to leverage the benefits of both security and cost-effectiveness to enhance the IT security framework.
Shamir's Secret Sharing scheme involves two phases: a sharing phase and a reconstruction phase. For each phase, we present a new implementation based on the Map Reduce design model. Consequently, each phase of the scheme incorporates Map and Reduce steps.

4.2.2 Data Sharing

Our approach integrates the Secret Sharing scheme with the MapReduce architectural pattern. Shamir's Secret Sharing (1979) is a cryptographic algorithm that divides a secret into parts, distributing these parts to participants, where a subset of these parts is required to reconstruct the original secret. Map Reduce, developed by Google, is a framework for processing large datasets in parallel and often distributed computations.
Our approach leverages the advantages of secret sharing schemes while significantly improving performance in terms of execution time. The proposed Map Reduce approach involves dividing the secret (a file) into p parts, distributing these shares to cluster machines, and executing a secret sharing method to calculate shares of each part. The shares are then sent to storage servers. Figure 2 illustrates the data sharing phase of the proposed approach.

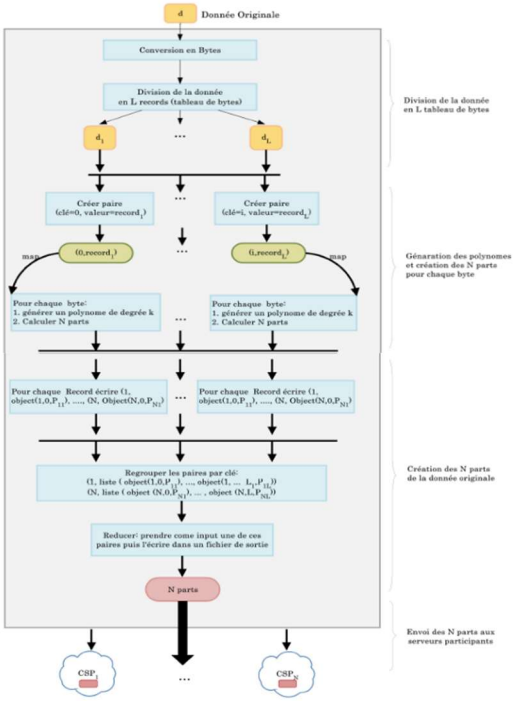

Figure 2: Data sharing phase based on the Map Reduce paradigm

In the Map step, large data is subdivided into several blocks, which are then sent to the slave nodes. Each node calculates the polynomials associated with the corresponding received data block. The first step in this phase is to divide the file to be shared according to the record size L chosen during the initialization phase. The data is divided into L records as an array of bytes. Each record is paired with its respective position in the original file to form pairs (key: record position in the file, value: record in the form of a byte array), which serve as the input for the Map functions.
For each byte of a record, the Map function randomly generates the coefficients of a polynomial P with the constant term equal to $byte_i$. After generating each polynomial, N points are computed, representing the N parts of $byte_i$. Upon completion of each mapper's execution, the output is represented as follows: (key = i , (value = i , position, record-$part_{ij}$)).
After all Map tasks are completed, a tri-grouping is performed on the returned pairs, and the new pairs are passed to a Reduce task. This sorting and grouping process involves sorting the pairs according to their keys and assembling pairs with the same key into the same group.
Each group of pairs (key, list (values)) is sent to a Reducer. Each Reducer processes one of these pairs as input and writes it to an output file. Consequently,

we will have multiple files, each representing a part of the original file.

4.2.3 Data Reconstruction

Similar to the data sharing process, we assume that the shares are stored on HDFS. To reconstruct the original file, we use K parts. Figure 3 illustrates the data reconstruction process based on the MapReduce paradigm.

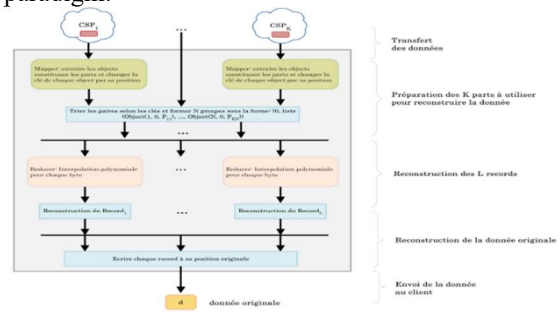

Figure 3: Data reconstruction phase based on the Map Reduce paradigm

The first step in this phase is to read the K parts and send each part to a different mapper. Each mapper will extract the objects constituting the parts and change the key of each object to its position.

After all Map tasks are completed, a sorting operation is performed on the pairs, and the resulting pairs are sent to a Reduce task. At this stage, the pairs are sorted according to their keys and then assembled into groups. This results in N groups, each of which is sent to a Reducer. Here, the keys are the positions of the records in the original file, with each share having a key (ID) that belongs to the interval [0, fileSize - 1]. Thus, shares with the same key represent shares of the same record at a certain position in the original file.

The role of each Reducer is to rebuild a record to eventually reconstruct the original file. Each pair passed to the Reducer contains the shares from 1 to K of a particular record. For example, consider the first pair, which contains the K parts (byte arrays of the same size = sizeRecord) of the first record. These byte arrays are traversed by byte, and the first bytes are extracted from the K arrays. Polynomial interpolation is then used to find the polynomial of degree k-1 that passes through all these points. We use Lagrange interpolation to find the unique polynomial P(x) of degree k-1. The constant term of the found polynomial represents a byte of a record from the original file.

First, we construct the Lagrange polynomials using the formula: $L_{(n,\,j)}(x) = (i = 0,\ ij)3\ \ ((x - x_i))/((x_i - x_j))$.

Then, we can deduce the Lagrange interpolation polynomial p(x) by: $p(x) = (j = 0)^{(k-1)} y_j L^{(k-1,\,j)}(x)$.

Obviously, the constant term of the polynomial is a byte of a record of the original file. The Reducer will perform these steps for each of the bytes of the k parts. At the end of each Reduce function, a byte array will be generated and will constitute the record j of the file; this record is then written to its original position and the original file is possibly correctly rebuilt.

5. PERFORMANCE ANALYSIS

Our approach leverages the Secret Sharing scheme to create a secure, fault-tolerant data storage service for collaborative work environments, including Cloud environments. These perfect sharing schemes encode data into shares such that only specific valid combinations of those shares can reconstruct the original data, while invalid combinations reveal no information about the encoded data. By storing these shares across different servers, the encoded data remains confidential as long as a sufficient number of servers are not compromised. However, most perfect secret sharing schemes suffer from computational inefficiencies. We address this issue by introducing a MapReduce version of the secret sharing scheme.

Our proposed approach combines a perfect partition schema with the MapReduce paradigm, significantly enhancing computational performance and achieving higher speeds compared to standard secret sharing schemes. In this section, we evaluate the properties and performance of the proposed system to demonstrate that combining a perfect partition scheme with the MapReduce paradigm can build reliable and secure distributed data storage systems for Cloud computing environments.

5.1 Experimental Analysis

5.1.1 Experimental Setup

To evaluate our approach, we installed a multinode cluster using Cloudera CDH 5. This setup aims to test, compare, and evaluate the performance of various secret sharing algorithms implemented using the MapReduce approach. Apache Hadoop, developed in Java, is the most well-known framework for implementing MapReduce.

Several Hadoop distributions facilitate the installation and management of a Hadoop cluster. Among these, Cloudera is the most widely deployed, offering robust tools for deployment, management, and monitoring. Thus, we chose Cloudera for our platform.

Cluster Technical Characteristics: We implemented the cluster using three virtual machines. One machine was dedicated to the NameNode/DataNode (which also hosts all administrative services), and

the other two were used as simple DataNodes. The physical machine used had 32 GB of memory and 2 TB hard disk capacity. For virtualization, VMware Workstation was used. Each virtual machine ran CentOS 7, with 8 GB of memory, 500 GB of disk space, and 4 vCPUs.

5.1.2 Data Sharing

We conducted experiments comparing Shamir's Secret Sharing Scheme (SSSS) with its MapReduce version (mrSSS). The criteria for evaluating the performance of each scheme were:

- Execution Time: Time required creating and reconstructing shares, measured in seconds.
- Scalability: Performance impact with increasing data sizes and thresholds.
- Resource Utilization: Efficiency in utilizing computational and memory resources.

The results demonstrate the strengths and weaknesses of each scheme in different application scenarios, as illustrated in Figures 4-7.

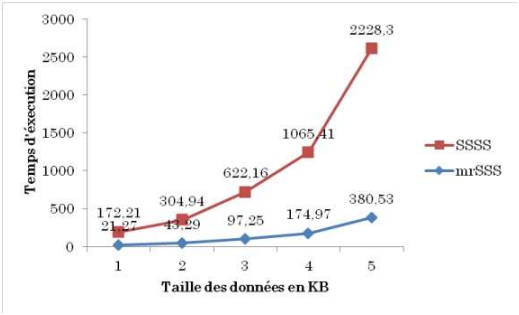

Figure 4: Sharing phase for different data sizes (N = 5, K = 2)

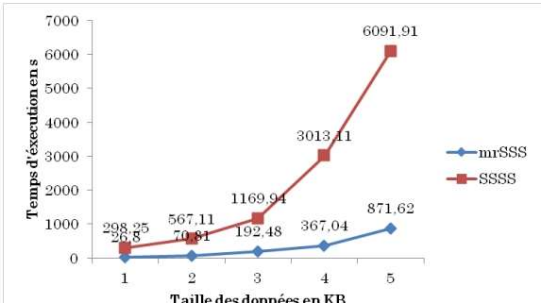

Figure 5: Sharing phase for different data sizes (N = 10, K = 4)

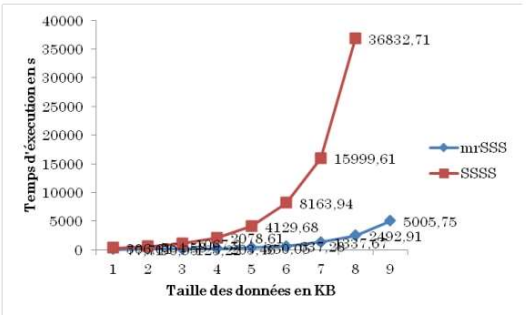

Figure 6: Sharing phase for different data sizes (N = 10, K = 6)

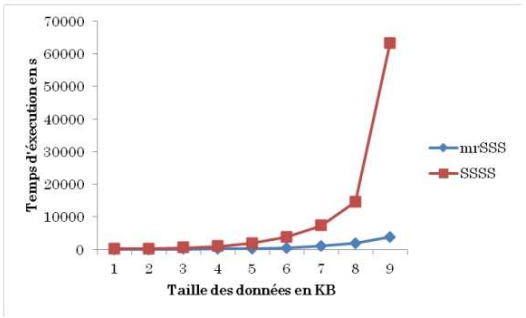

Figure 7: Sharing phase for different data sizes (N = 5, K = 3)

The experiments aimed to show the impact of data size variations on the performance of each secret sharing scheme (SSS) in terms of data sharing. Data sizes from 2 KB to 262 KB were evaluated. The generated data was arbitrary, as the evaluations were not specific to any particular domain where SSS algorithms might be applied.

We presented four primary sets of results using configurations of (N = 5, K = 2), (N = 10, K = 4), and (N = 6). Here, N refers to the number of shares created, and K refers to the number of shares required for reconstructing the original data using each SSS algorithm. The time required for creating and reconstructing shares is reported in seconds.

5.1.3 Data Reconstruction

Similar to data sharing, we compared the performance of data reconstruction for each scheme by varying thresholds and data sizes. The criteria for evaluating data reconstruction were:

- Reconstruction Time: Time required to reconstruct data from shares, measured in seconds.
- Accuracy: Correctness of the reconstructed data.
- Performance under Load: Efficiency with different configurations of shares and data sizes.

Figures 8-11 present the results.

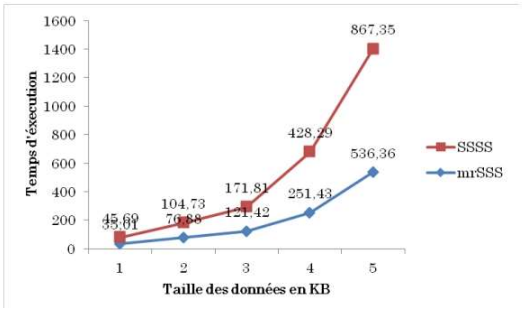

Figure 8: Reconstruction phase for different data sizes (N = 5, K = 2)

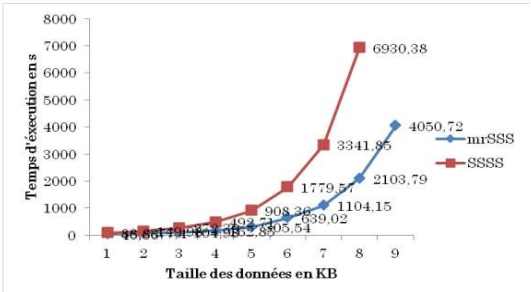

Figure 9: Reconstruction phase for different data sizes (N = 10, K = 4)

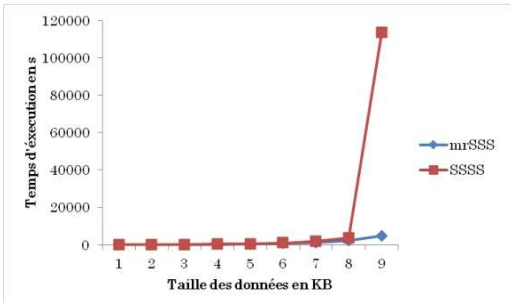

Figure 10: Reconstruction phase for different data sizes (N = 10, K = 6)

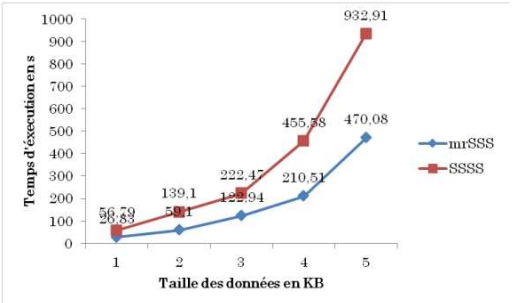

Figure 11: Reconstruction phase for different data sizes (N = 5, K = 3)

We evaluated four main series of results using configurations of (N = 5, K = 2), (N = 10, K = 4), and (N = 5, K = 3). Here, N refers to the number of shares (fragments) created, and K refers to the number of shares required to reconstruct the original data using each SSS algorithm. The execution time required for creating and reconstructing shares is reported in seconds.

From Figures 4-11, it is evident that mrSSS is the fastest algorithm regardless of data size. Notable observations include the scalability issues of SSSS with increasing data sizes; beyond 131,072 KB, shares can no longer be created or rebuilt as the test machine encounters out-of-memory exceptions. Furthermore, increasing the parameters shows that the sharing phase generates a significant increase in performance time.

5.2 Security Analysis

5.2.1 Data Confidentiality

During data fragmentation, the original data d is transformed into N unrecognizable fragments, each stored in a different physical location. Data reconstruction is only possible when k out of these fragments are collected. The security of the resulting data primarily depends on the parameters k and N, which define the extent of dispersion. A higher value of k makes it more difficult for an attacker to access at least k storage locations. When N is close to k, availability increases, but it also limits the choice of fragments, making data retrieval more challenging. If k equals N, all fragments are necessary for data reconstruction.

In case an intruder manages to steal shares from x <= K Cloud providers, the probability of reconstructing the original data depends on:

1. The user-defined value of k. A higher k lowers the probability of data reconstruction.
2. The number x of stolen shares. The probability of reconstructing the data increases with x. However, mrSSS remains secure because recovering shares from at least k Cloud Service Providers (CSPs) simultaneously is highly challenging.

5.2.2 Data Availability

Data availability is inherently ensured by the Secret Sharing scheme. Our approach guarantees that a user can reconstruct their secret if k or more CSPs are honest and their shares are accessible. It is crucial to choose k appropriately to ensure data availability.

5.2.3 Data Integrity

Dispersed fragments can be modified, especially when stored on unreliable devices. Ensuring data integrity can be achieved in several ways:

- Cryptographic Hashing: Adding a hash to the data before fragmentation allows verification of data integrity by comparing the hashes after defragmentation.
- Voting System: Replicating fragments across multiple nodes allows comparison of

multiple replicas during defragmentation, with the most frequent answer being selected. This method, while faster, is less efficient in terms of storage and reduces the protection offered by dispersion due to increased exposure of replicated data.

Our proposed approach does not currently address long-term security components, which will be considered in future work.

5.3 Time Complexity

Time complexity helps to understand the factors influencing execution time. This section describes the time complexities of data-sharing and reconstruction processes from the user perspective. We conducted 100 test cases with 1 GB of data, modifying the N and k parameters.

5.3.1 Data Sharing

For the conventional Secret Sharing scheme, sharing data involves generating a random polynomial, computing N points, and distributing them among participants. The time complexity is O(N). In the MapReduce version, data is first transformed into a set of bytes and divided into blocks of size Lbefore being processed by the Map functions. Each Map function computes L random polynomial equations, distributing the time complexity among simultaneously executed Map functions.

For instance, the execution times for sharing 1 GB of data with mrSSS are plotted in Figures 12 and 13 with respect to k and N. The execution time is approximately 15 seconds when N = k = 3. Figure 12 shows that execution time increases rapidly with k when N = k. Figure 13 illustrates that execution time increases linearly with N when k = 4. Higher k values affect execution time more significantly than higher N values.

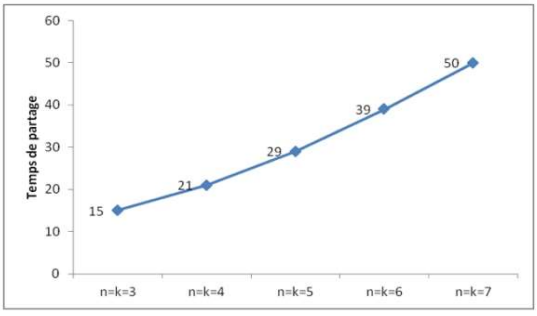

Figure 12: Data Sharing Time with respect to K

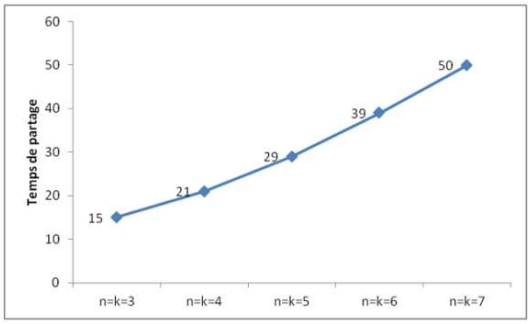

Figure 13: Data Sharing Time with respect to N

5.3.2 Data Reconstruction

To reconstruct data, we use polynomial interpolation to find the polynomial of degree k-1 that passes through the k points. Lagrange interpolation is used to find the unique polynomial. The constant term of the polynomial is the original datum. In the MapReduce version, these steps are repeated according to the data size, resulting in a time complexity of O(k log k).

For example, the execution time for reconstructing 1 GB of data is plotted in Figure 14 with respect to k. The execution time is about 10 seconds when k = 3 and increases polynomially with k.

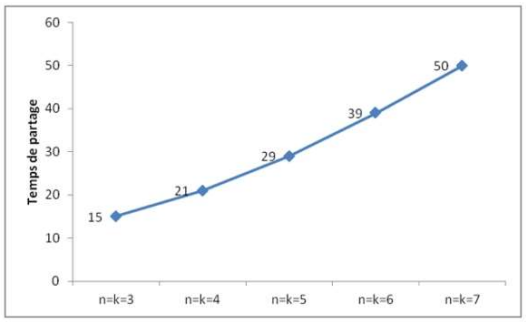

Figure 14: Data Reconstruction Time with respect to K

6. COMPARATIVE ANALYSIS AND CONTRIBUTIONS

In this section, we outline the key contributions of our research and compare them with existing solutions in the literature. Our focus is on how our proposed approach using the MapReduce version of the Secret Sharing scheme (mrSSS) stands apart from and advances the state of the art.

6.1 Key Findings

- Enhanced Computational Efficiency: Unlike traditional Secret Sharing schemes, which often suffer from computational inefficiencies, our approach integrates a MapReduce paradigm. This integration significantly enhances computational

performance, allowing for faster data sharing and reconstruction processes. Previous works demonstrate that while Secret Sharing provides strong security guarantees, their implementations often struggle with scalability and performance issues, especially as data sizes increase.

- Scalability Improvements: Our method addresses scalability challenges more effectively. Existing solutions, including Shamir's Secret Sharing Scheme (SSSS) and its variants, show performance degradation with large data sizes and high thresholds. In contrast, our MapReduce version (mrSSS) maintains efficiency even with larger datasets, overcoming the memory and processing limitations observed in prior research.
- Performance Benchmarks: Our experimental results indicate that mrSSS provides superior performance in both data sharing and reconstruction phases compared to conventional SSSS. For example, the execution time of mrSSS is significantly lower for various data sizes and thresholds, as illustrated in Figures 12-14. This improvement is a notable advancement over traditional schemes, which often show exponential increases in execution time with growing data sizes.
- Practical Implementation: The use of Cloudera CDH 5 and Hadoop for implementing our approach provides a practical framework for deploying and managing MapReduce-based Secret Sharing systems. While previous studies have explored theoretical models and simulations, our work provides a concrete implementation that can be readily adapted for real-world applications.
- Security and Confidentiality: Our approach not only improves performance but also upholds high security standards. We ensure that data remains confidential and secure even when shares are distributed across multiple servers. Prior research has sometimes compromised security for performance, but our approach maintains a balance by ensuring that security remains robust while optimizing computational efficiency.

6.2 Comparison with Prior Literature

Prior Work: Traditional Secret Sharing schemes, such as those developed by Shamir and Blakley, offer theoretical foundations for secure data sharing. However, these schemes often face limitations related to computational inefficiency and scalability. Studies by [13] and [14] highlight these inefficiencies and the challenges faced in real-world deployments.

Our Contribution: By introducing the MapReduce-based approach, we provide a solution that not only addresses the performance limitations but also scales effectively with increasing data sizes and complexity. Our work builds on the theoretical foundations laid by earlier researchers and translates them into a practical, efficient, and scalable solution. This advancement is a significant departure from earlier methods, making our approach more suitable for modern, data-intensive applications.

In summary, our research advances the field by offering a more efficient and scalable solution to the challenges of Secret Sharing in Cloud environments. We bridge the gap between theoretical models and practical applications, setting a new benchmark for performance and security in distributed data storage systems.

7. CONCLUSION

Cloud Computing is a global concept with no geographical boundaries. The computers used to process and store user data can be located anywhere worldwide, depending on the capabilities required by the global computer networks used for cloud computing. The security of a cloud computing system is a contentious issue that can hinder its adoption. Many may argue that controlling a private cloud on-site is safer than having resources managed externally. However, the essence of cloud-based services, whether private or public, lies in the external management of provided services. This creates a significant incentive for cloud service providers to ensure robust and secure management of their services.

Secret sharing schemes are being applied to cloud computing environments because they can distribute data to multiple servers, preventing system failures due to natural disasters or human errors. Additionally, these schemes do not provide any information about the data if the number of shares is below the threshold, making the system secure even with the least computational security. Secret sharing schemes protect information against malicious insiders and attackers by distributing data across multiple servers in different locations. Such systems are essential security mechanisms, as they force adversaries to compromise multiple sites to learn or alter the data. However, these methods have three

major limitations: the long-term security of long-lived data, the assumption that data corruption occurs only at the time of recovery, and degraded computational performance with large data.

To address these concerns, we propose two new approaches to secure data storage in cloud computing environments: pSSS and mrSSS. pSSS addresses the long-term data security issue and ensures the correctness of data storage even when some servers fail. This is achieved through cryptographic techniques, such as distributed digital signatures and threshold cryptography-based key management. mrSSS implements Secret Sharing schemes according to the MapReduce architecture paradigm, combining security and cost-effectiveness to provide a more comprehensive IT security framework and potentially improve computational performance for large data.

Performance tests have demonstrated that our approach is more efficient regardless of the data size. The results also highlight scalability issues with the SSSS algorithm concerning data size, with significant performance increases only observed during the data sharing phase.

This work can be extended by combining the two proposed approaches into a single, secure distributed data storage system for cloud computing environments. Additionally, it would be beneficial to present the MapReduce version of other secret sharing schemes, such as Proactive Secret Sharing and Verifiable Secret Sharing, to further improve the system's performance. Another potential improvement is a broader deployment. A wider and more practical deployment of our system at a public cloud provider would allow for a comprehensive evaluation of the technique with real data on cloud usage and customer demands.

REFERENCES:

[1] Chen, Yanpei and Paxson, Vern and Katz, Randy H, "What's new about cloud computing security," University of California, Berkeley Report No. UCB/EECS-2010-5, January 2010, Page 5.

[2] Sotto, Lisa J and Treacy, Bridget C and McLellan, Melinda L, "Privacy and Data Security Risks in Cloud Computing," World Communications Regulation Report, 5, 2010, Page 38.

[3] Hubbard, Dan and Sutton, Michael and others, "Top threats to cloud computing v1.0," Cloud Security Alliance, 2010.

[4] Meer, Haroon and Arvanitis, Nick and Slaviero, Marco, "Clobbering the cloud," Black Hat USA, 2009, Page 197.

[5] Gonzalez, Nelson and Miers, Charles and Redigolo, Fernando and Simplicio, Marcos and Carvalho, Tereza and Naslund, Mats and Pourzandi, Makan, "A quantitative analysis of current security concerns and solutions for cloud computing," Journal of Cloud Computing: Advances, Systems and Applications, 1, 2012, Page 11.

[6] Balduzzi, Marco and Zaddach, Jonas and Balzarotti, Davide and Kirda, Engin and Loureiro, Sergio, "A security analysis of Amazon's Elastic Compute Cloud service," Proceedings of the 27th Annual ACM Symposium on Applied Computing, 2012, Pages 1427–1434.

[7] Bugiel, Sven and N¨urnberger, Stefan and P¨oppelmann, Thomas and Sadeghi, Ahmad-Reza and Schneider, Thomas, "AmazonIA: When elasticity snaps back," Proceedings of the 18th ACM Conference on Computer and Communications Security, 2011, Pages 389–400.

[8] Xiao, Zhifeng and Xiao, Yang, "Security and privacy in cloud computing," IEEE Communications Surveys & Tutorials, 15, 2013, Pages 843–859.

[9] Bhadauria, Rohit and Sanyal, Sugata, "Survey on security issues in cloud computing and associated mitigation techniques," arXiv preprint arXiv:1204.0764, 2012.

[10] Rai, Rashmi and Sahoo, Gadadhar and Mehfuz, Shabana, "Exploring the factors influencing the cloud computing adoption: A systematic study on cloud migration," SpringerPlus, 4, 2015, Page 197.

[11] Chang, Victor and Ramachandran, Muthu, "Towards achieving data security with the cloud computing adoption framework," IEEE Transactions on Services Computing, 9, 2016, Pages IEEE Transactions on Services Computing.

[12] Xiao, Zhifeng and Xiao, Yang, "Security and privacy in cloud computing," IEEE Communications Surveys & Tutorials, 15, 2013, Pages 843–859.

[13] Rai, Rashmi and Sahoo, Gadadhar and Mehfuz, Shabana, "Exploring the factors influencing the cloud computing adoption: A systematic study on cloud migration," SpringerPlus, 4, 2015, Page 197.

[14] Yang, L., Wu, Q., & Zhang, F., "A secure and efficient cloud data storage scheme using verifiable secret sharing," Future Generation Computer Systems, 108, 2020, Pages 1-12.

[15] Chen, J., Liu, Y., & Li, X., "Hybrid cloud security model combining secret sharing and homomorphic encryption," Journal of Cloud Computing, 10(1), 2021, Pages 1-18.