International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

Integrity Verification Algorithm for Cloud-Stored Documents

Mohd Tajammul
Department of Computer Science
Sharda University
Greater Noida, India

Manish Adawadkar
Kelley School of Business
Indiana University, Bloomington, USA

Raziullah Khan
Department of Computer Science
Rajiv Gandhi Proudyogiki
Vishwavidyalaya, Bhopal, India

***Abstract*—Those without the necessary tools can nonetheless succeed in the business and marketing world of today. The sole reason for this is cloud computing. Many firms are still undecided about whether or not to use cloud computing. Their most significant obstacle is the protection of their private information. This research paper is an effort to persuade those businesses or individuals who are still considering using this most effective, affordable, and best service. This study presents an integrity testing algorithm that verifies entire documents at the character level both before they are uploaded to cloud storage and after they are downloaded. There are similarities between the algorithm's concept and "error detection and correction."**

***Keywords*—Cloud Storage; Integrity; Cloud Compuitng; Encryption, Decryption.**

I. INTRODUCTION

After several decades of advancement, cloud computing has matured into a sophisticated and multifaceted paradigm. Rather than representing a single innovation, it embodies the integration of multiple technologies, with the Internet serving as its foundational element. Complementary components such as virtualization, distributed computing, and parallel processing further enhance its capabilities. A major advantage of this technology is its capacity for remote accessibility, enabling users to access services at any location and time through a pay-per-use framework. Leveraging virtually limitless resources from cloud service providers allows individuals and organizations to meet deadlines and comply with service-level agreements efficiently.

The conceptual roots of cloud computing can be traced to 1969, when Professor John McCarthy proposed that computers might eventually be organized as a public utility, similar to the telephone network of his time. He suggested that such a “computer utility” could form the foundation of a transformative industry. This vision aligns with the ideas expressed by Douglas Parkhill in his 1966 work The Challenge of the Computer Utility, in which he outlined core attributes such as elasticity, resource pooling, and shared infrastructure.

In contemporary terms, the National Institute of Standards and Technology (NIST) defines cloud computing as “a concept for offering convenient, on-time network access to a shared pool of quickly provided and released computer resources that may be configured (such as servers, networks, storage, apps, and services) with little administration endeavor or communication with the service provider” [11–12]. According to this model, three service types, four deployment strategies, and five essential characteristics collectively support high availability and operational flexibility [11–12]. The concept of cloud storage security has been sketched in Fig. 1.

A. Cloud Computing Models

- Private Cloud - This type of cloud model is only available to one specific company or to an individual who wants to store sensitive or secure data on it. An organization or a specific person owns this model. Only the owner of this cloud can utilize its full capacity [19].
- Public Cloud - One of the most basic types of cloud computing is this one. The public can access it. Use of the public cloud is recommended if the data we wish to put on cloud storage is not more sensitive [8].
- A hybrid cloud - is created by fusing the characteristics of two or more different clouds. This is advised in cases where a particular kind of cloud cannot satisfy customers' needs. This cloud's ability to accommodate different customer needs that go beyond what a single cloud can offer is an advantage [19].
- Community Cloud - This kind of cloud is referred to as a community cloud if it is developed by a few reliable parties or organizations and only these businesses are able to utilize its entire capacity [18].

B. Cloud Computing Services

- The software-as-a-service model This is cloud computing's most popular service model. In this paradigm, the cloud user has access to the program. The client can utilize this software, but they are unable to tamper with it because it is entirely maintained by the supplier. SaaS is offered by a number of businesses in the industry. These include IBM, Oracle, SalesForce.com, Microsoft, and NetSuite [18].
- The platform as a service: In cloud architecture, this paradigm is regarded as the intermediate model. Those who wish to launch their firm without having the necessary capital might use this model. equipment. This concept provides customers or users with a platform that they may use to build their purposes. A software engineer, for instance, wishes to start his own software development business but only has laptops and no suitable platform to create software on. Cloud SaaS is accessible to these individuals. Microsoft and Azure GAE are a couple of the businesses that offer PaaS [8].

- A service-based infrastructure: When it comes to cloud computing architecture, this approach is the lowest. With this service model, the client has pay-per-use access to a reserved infrastructure. For instance, if a company wishes to store its data outside of its boundaries, it can use cloud storage as a service. Similarly, if an individual wishes to carry out a computation that exceeds the capabilities of a single node, he can use cloud computing as a service. Under the terms of the Service Level Agreement (SLA), both of them will receive service and be billed on a pay-per-use basis. These include GoGrid, Flexiscale, Joyent, and Rackspace, among other well-known IaaS providers [19].

C. Cloud Storage Higly Addressale Issues

- Encrypting data while it's in transit and at rest
- Identity management and access control are inadequate.
- Incorrectly constructed storage containers or buckets
- APIs and interfaces that are not secure
- Data loss as a result of inadvertent overwriting or deletion
- Absence of monitoring and audit trails
- Risks shared by tenants in multi-tenant settings
- Poor data deletion procedures
- Shadow IT and unauthorized storage services
- Gaps in regulations and compliance

Figure 1: Encrypted cloud storage[2]

D. Contribution

Our contributions in this study can be summarized into four main points. First, we designed and implemented an algorithm that relies on a matrix-based approach rather than conventional group-based methods. The proposed algorithm automatically generates a matrix from the given input file, after which the data can be uploaded to the cloud—either in plaintext or encrypted form—using the Advanced Encryption Standard (AES). To the best of our knowledge, no existing method tests data integrity using this type of matrix-driven, nonlinear approach; thus, this work represents the first attempt in this direction.

The algorithm functions by producing a matrix both before uploading and after downloading the file from cloud storage. This matrix serves as a proof that the file has not been altered during transmission between the local desktop and the cloud server. Next, AES encryption is applied to the matrix-generated file to further enhance security. While files can be transmitted in plaintext, encrypting them is a widely accepted best practice to safeguard against unauthorized access, especially when data crosses security boundaries.

Third, we have developed a conceptual architecture for the proposed system to simplify understanding. This architecture clearly illustrates the secure and efficient flow of data between the end-user and the cloud. Lastly, we have carried out performance evaluations of our algorithm, measuring the execution time for both matrix generation and integrity testing. For small datasets, both AES and DES were applied to assess encryption and decryption performance. Experiments were conducted on files ranging from 500 KB to 5000 KB from two different sources. Table 1 presents the dataset characteristics, including file sources and sizes.

The remainder of this paper is organized as follows: Section 2 presents the motivation, Section 3 outlines the research background and literature gap, Section 4 details the proposed algorithm, Section 5 explains the data collection process, Section 6 discusses experimental results, Section 7 concludes the work, and Section 8 outlines potential directions for future research.Literature survey

II. LITERATURE SURVEY

Hassan Rasheed [4] introduced a security auditing framework for cloud computing, dividing the process into two main domains: data auditing and infrastructure auditing. Mohd. Tajammul, Rafat Parveen, and Mohd. Shahnawaz [8] examined a variety of cloud security challenges and related concerns. Subhashini and V. Kavitha [9] conducted a comprehensive review of security risks across different cloud service models, identifying 14 issues specific to SaaS, several in PaaS, and the remainder in IaaS, alongside proposed countermeasures and assessment techniques. Manas MN et al. [11] explored cloud computing hurdles and mitigation methods within the SaaS, PaaS, and IaaS frameworks, also focusing on VM-level memory and cache isolation for multi-tenant environments. In [18], Mohd. Tajammul and Rafat Parveen summarized ten critical standards under the Information Security Management System (ISMS). Zuojie et al. [3] devised a keyword-based search system for multi-tenant setups, enabling encrypted file retrieval from cloud storage. Liefi Wei et al. [1] developed the SecCloud and SecHDFS systems, ensuring encryption before cloud upload and decryption prior to computation, without involving the service provider. Laurace T. Yang et al. [5] proposed the parallel block GNFS and Weidman algorithm to enhance RSA security, discussing its constraints and GNFS's potential to factor extremely large integers. Dimitrias Zissis and Dimitrias Lekkas [10] presented a tabular summary of cloud security issues, covering protection levels, requirements, and potential threats. Den Bonch and Mathew Franklin [6] proposed an identity-based encryption scheme leveraging Weil pairing over bilinear maps, outlining practical applications. Manish M. Potey et al. [11] suggested a homomorphic encryption method enabling computations on encrypted data in public clouds, storing the outcomes locally. P. Ravi Kumar et al. [14] investigated data security risks along with their mitigation strategies. Ali Azougaghe et al. [7] designed a hybrid encryption technique—AES for securing data and ElGamal for protecting the AES key—retaining the key locally while uploading encrypted data. Tim Mather and Subra Kumaraswamy [19] reviewed various facets of cloud security, including infrastructure protection, data and storage safeguarding, identity and access controls, key exchange mechanisms, and associated issues. Micheal Armbrust [12] provided an overview of unlocking the benefits of cloud computing while mitigating its drawbacks. Hsun Chuhang et al. [15] introduced a privacy-preserving approach that balances user confidentiality with system efficiency. Mahindha [17] implemented dual-layer encryption by applying DES to plaintext, followed by RSA on the DES output. Qian Wang et al. [16] explored public auditing and dynamic data handling in cloud storage, emphasizing integrity and adaptability. Finally, Naga Malleswari TYJ and Vadivu G [22] studied VM deduplication, assessing its implications for data security breaches.

III. RESEARCH GAP

The literature survey indicates that while extensive research has been conducted on encryption, decryption, and authentication techniques, document integrity verification at the character level remains unexplored. No existing algorithm is capable of performing such verification both prior to uploading data to cloud storage and after its retrieval. This gap underscores the need for developing an algorithm that ensures character-by-character integrity validation.
This study is framed within the IaaS (Infrastructure as a Service) model, acknowledging that most IaaS platforms are comparatively less secure than conventional desktop environments. The goal is to improve the security and reliability of IaaS or HaaS (Hardware as a Service—encompassing both computation and storage) by minimizing hacking risks and enabling accurate integrity assessment of stored data.
To address this, the research focuses on the following key questions:

1. How can users verify that data uploaded to a cloud server remains intact and unaltered?
2. If alterations occur, how can users pinpoint the specific characters or sections affected?
3. How can the security of files in multitenant cloud storage be enhanced?

This work seeks to provide solutions to these questions and close the identified research gap.

IV. PROPOSED METHOD

The proposed algorithm is founded on a simple yet effective concept. Imagine a scenario where a user secures cloud storage space—possibly in encrypted form—but lacks complete trust in the cloud service provider (CSP), its current staff, or former employees. Owing to their access privileges, these parties could potentially compromise security, even though doing so may not be straightforward. If a user rents cloud storage, uploads data, and at some point a breach occurs altering that data—particularly if it is highly sensitive such as financial or military information—an important question emerges: How can the owner detect if the stored data was tampered with while in the cloud?
To resolve this, we developed an algorithm that tests data integrity before upload. It produces a matrix capturing the frequency of each character in the document. The data, whether in plain text or encrypted form, is then uploaded. Upon retrieval, the same algorithm is run to generate a new frequency matrix. Comparing the two matrices reveals any discrepancies, thereby identifying unauthorized changes. Our implementation considers characters ‘a’ to ‘z’ and digits ‘0’ to ‘9’. The frequency of ‘a’ occupies the first matrix position, ‘b’ the second, and so on, with ‘9’ in the 36th position. The resulting matrix has a fixed size of 6×6. Although storing data in plain text on the cloud is possible, encryption is strongly recommended due to the multi-tenant nature of cloud systems, where multiple clients share hardware resources like storage and processing power. This environment increases the likelihood of data leakage. For instance, computational tasks may leave residual information in server caches, which could be accessed by subsequent users of the same machine. Prior to cloud upload, it is crucial to assess the potential risks from unauthorized access. The main threat actors include:

- The CSP itself
- Internal CSP employees
- Other tenants in a multi-tenant environment or attackers conducting man-in-the-middle (MITM) attacks

Because CSPs and their staff hold extensive system privileges, these entities present significant risks if sensitive data is stored without encryption. The concept of proposed scheme has been sketched in Fig. 2.

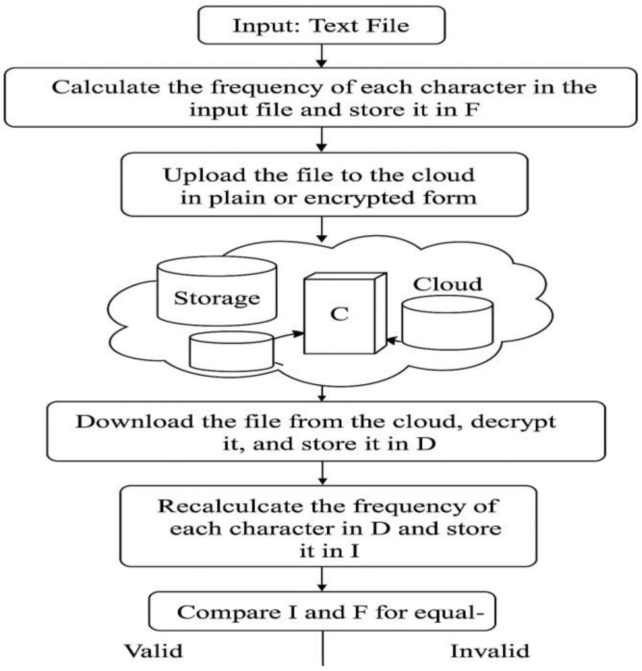

Input: Text File
Calculate the frequency of each character in the input file and store it in F
Upload the file to the cloud in plain or encrypted form
Storage
C
Cloud
Download the file from the cloud, decrypt it, and store it in D
Recalculcate the frequency of each character in D and store it in I
Compare I and F for equal-
Valid
Invalid

Figure 2: Conceptual diagram of proposed approach

A. Proposed Algorithm for the given approach

Algorithm 1 — Pre-Upload Profiling

Input: Document D (bytes or text), mode ∈ {plain, encrypted}
Output: Baseline frequency matrix M_0 (6×6), optionally its checksum σ

```
1. Initialize a zero vector F[0..35] ← 0.
2. For each byte/char x in D:
        •If x is a letter (A–Z/a–z), set y ← toLower(x).
        •If y ∈ a..z, increment F[pos(y)] ← F[pos(y)] + 1
        •Else if x ∈ 0..9, increment F[pos(x)] ← F[pos(x)] + 1
        •Otherwise: ignore.
3. Convert F to a 6×6 matrix M₀ by filling row-major: $M_0[\lfloor i/6 \rfloor, i \bmod 6] \leftarrow F[i]$ for i = 0..35.
4. (Optional but recommended) Compute checksum σ ← H(F) using a cryptographic hash (e.g., SHA-256) or HMAC with a user-held key.
5. Store M₀ (and σ, if used) **locally** or in a trusted channel.
6. Upload D to the cloud in the chosen mode (plain or encrypted).
```

Algorithm 2 — Post-Download Verification

Input: Downloaded document D′, baseline M_0 (and σ if used)
Output: Verdict tampered ∈ {true,false}, and difference report Δ

```
1. Recompute frequency vector F′ over D′ exactly as in Steps 1–2 of Algorithm 1.
2. Form matrix M₁ from F′ as in Step 3 of Algorithm 1.
3. (If checksum used) Verify H(F′) == σ. If mismatch, set tampered ← true and report **checksum mismatch**; stop (or continue to Step 4 to quantify deltas).
4. Compute difference matrix $\Delta \leftarrow M_1 - M_0$.
5. If any entry of $\Delta \neq 0$, set tampered ← true; else tampered ← false.
6. Return (tampered, Δ).
```

Comparison Function (for reporting)

- Let $L1 = \Sigma |\Delta[i,j]|$ (total count drift).
- Let $L\infty = \max |\Delta[i,j]|$ (max per-symbol drift).
- Report indices (or characters) where $\Delta \neq 0$, e.g., (char, old, new, delta).

B. Mathematical Description of the proposed approach

Definitions

1. Let the **allowed symbol set** be:
Σ={a,b,…,z,0,1,…,9}\Sigma = \{ a,b,\dots,z,0,1,\dots,9 \}Σ={a,b,…,z,0,1,…,9}
where ∣Σ∣=36|\Sigma| = 36∣Σ∣=36.

2. Define an **index mapping function**:
ϕ:Σ→{0,1,…,35}\phi: \Sigma \rightarrow \{0,1,\dots,35\}ϕ:Σ→{0,1,…,35}
such that:
ϕ(a)=0, ϕ(b)=1,…,ϕ(z)=25, ϕ(0)=26,…,ϕ(9)=35\phi(a) = 0,\ \phi(b) = 1, \dots, \phi(z) = 25, \ \phi(0) = 26, \dots, \phi(9) = 35ϕ(a)=0, ϕ(b)=1,…,ϕ(z)=25, ϕ(0)=26,…,ϕ(9)=35

3. *Let DDD be the document (string) before upload, and D'D'D' be the document after download.*
Let n=∣D∣n = |D|n=∣D∣ be the total number of characters in DDD.

4. Let the **frequency vector** *F∈N036F \in \mathbb{N}_0^{36}F∈N036 be defined as:*
Fk=∑i=1n1(Di∈Σ ∧ ϕ(Di)=k)F_k = \sum_{i=1}^{n} \mathbf{1}\big(D_i \in \Sigma \ \wedge\ \phi(D_i) = k \big)Fk=i=1∑n1(Di∈Σ ∧ ϕ(Di)=k)
where 1(⋅)\mathbf{1}(\cdot)1(⋅) is the indicator function (1 if condition is true, 0 otherwise).

5. Let the **matrix transformation** be:
M∈N06×6,Mr,c=F6r+cM \in \mathbb{N}_0^{6 \times 6}, \quad M_{r,c} = F_{6r + c}M∈N06×6,Mr,c=F6r+c
for r,c∈{0,1,…,5}r, c \in \{0,1,\dots,5\}r,c∈{0,1,…,5}.

Pre-Upload Profiling (Baseline Generation)

Given: Document DDD
Steps:
1. Compute baseline frequency vector:

Fk(0)=∑i=1n1(Di∈Σ ∧ ϕ(Di)=k)F^{(0)}_k = \sum_{i=1}^{n} \mathbf{1}\big(D_i \in \Sigma \ \wedge\ \phi(D_i) = k \big)Fk(0)=i=1∑n1(Di∈Σ ∧ ϕ(Di)=k)

2. Form baseline matrix:

Mr,c(0)=F6r+c(0)M^{(0)}_{r,c} = F^{(0)}_{6r + c}Mr,c(0)=F6r+c(0)

3. (Optional) Compute baseline checksum:

σ=H(F(0))\sigma = H\big(F^{(0)} \big)σ=H(F(0))
where HHH is a cryptographic hash (e.g., SHA-256).

Post-Download Verification

Given: Downloaded document D'D'D′, baseline M(0)M^{(0)}M(0)
Steps:
1. Compute post-download frequency vector:

Fk(1)=∑i=1n′1(Di′∈Σ ∧ ϕ(Di′)=k)F^{(1)}_k = \sum_{i=1}^{n'} \mathbf{1}\big(D'_i \in \Sigma \ \wedge\ \phi(D'_i) = k \big)Fk(1)=i=1∑n′1(Di′∈Σ ∧ ϕ(Di′)=k)
where n′=∣D′∣n' = |D'|n′=∣D′∣.

2. Form post-download matrix:

Mr,c(1)=F6r+c(1)M^{(1)}_{r,c} = F^{(1)}_{6r + c}Mr,c(1)=F6r+c(1)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

3. Compute **difference matrix**: Δr,c=Mr,c(1)−Mr,c(0)\Delta_{r,c} = M^{(1)}_{r,c} - M^{(0)}_{r,c}Δr,c=Mr,c(1)−Mr,c(0)
4. Decision rule: tampered={true,if ∃(r,c):Δr,c≠0false,otherwise\text{tampered} = \begin{cases} \text{true}, & \text{if } \exists (r,c) : \Delta_{r,c} \neq 0 \\ \text{false}, & \text{otherwise} \end{cases}tampered={true,false,if ∃(r,c):Δr,c=0otherwise
5. (If checksum used) Tampering is **true** if: H(F(1))≠σH\big(F^{(1)} \big) \neq \sigmaH(F(1))=σ

C. Data collection, description and ranges

To implement the algorithm, textual data was gathered from online sources: ttps://file-examples.com/index.php/text-files-and-archives the data size ranges from 10MB to 100MB. This data is purely in textual form and does not contains any special characters except '#' and '$'. This is the limitation of the proposed system to deal with the text data only, dealing with the images and audios or videos or any unstructured data is beyond the concept of this research. The given proposed has been tested and validated against the given data to verify the efficiency of the system against the security measures. One more feature of the given data is that the number of characters or words in each document can be recorded.

V. EXPERIMENTAL SETUP AND RESULTS DISUCSSION

For performance evaluation, we implemented the proposed algorithm in Java (JDK 1.6.0) using NetBeans IDE 7.2.1. Tests were conducted on a 64-bit Windows Pro machine equipped with an Intel Core i3-5005U @ 2.00 GHz, 8 GB RAM, and 1 TB HDD. Each experiment was executed in a single-threaded configuration and repeated ten times. The experimental results, presented in Table I, correspond to file sizes between 1 MB and 10 MB, where MCT represents Matrix Creation Time and ITT indicates Integrity Testing Time. As per Table II, files up to 10 MB were processed in 2 seconds, confirming that the algorithm is suitable for large-scale data, aligning with the requirements of cloud storage environments. A real execution of the system has been shown in a screenshot and attached in Fig. 3.

TABLE I: Experimental Results

Data Ranges	Matrix Creation Time	Integrity Testing Time
10MB	2 Seconds	1 Second
20MB	2 Seconds	1 Second
30MB	2 Seconds	1 Second
40MB	3 Seconds	1 Second
50MB	3 Seconds	1 Second
60MB	3 Seconds	1 Second
70MB	3 Seconds	1 Second
80MB	4 Seconds	1 Second
90MB	4 Seconds	1 Second
100MB	4 Seconds	1 Second

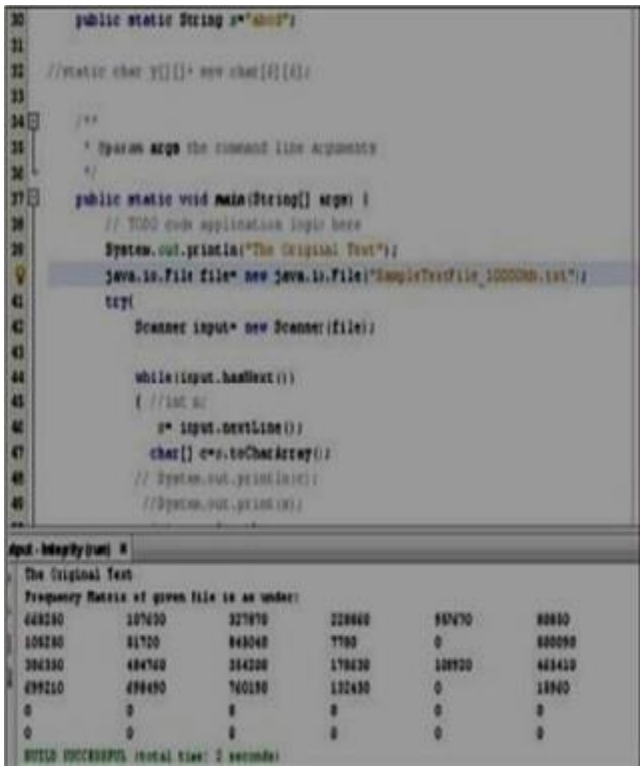
Step 2 of the algorithm specifies that files may be uploaded to cloud storage either as plain text or in encrypted form, with encryption being the recommended approach. At this stage, AES encryption was chosen due to the official withdrawal of DES in 2005, though DES encryption was still tested for reference. The AES encryption process was executed within the same system setup, using datasets between 10 MB and 100 MB. Encryption performance results are documented in Table II, whereas decryption outcomes are provided in Table III.

TABLE II: Encryption time computation by AES and DES

Data Ranges	AES Algorithm Time Computation	DES Algorithm Time Computation
10MB	60 Seconds	51 Second
20MB	97 Seconds	107 Second
30MB	121 Seconds	152 Second
40MB	144 Seconds	238 Second
50MB	180 Seconds	322 Second
60MB	320 Seconds	598 Second
70MB	374 Seconds	721 Second
80MB	427 Seconds	812 Second
90MB	589 Seconds	1093 Second
100MB	650 Seconds	1223 Second

TABLE III: Decryption time computation by AES and DES

Data Ranges	Matrix Creation Time	Integrity Testing Time
10MB	53 Seconds	51 Second
20MB	103 Seconds	107 Second
30MB	132 Seconds	152 Second
40MB	153 Seconds	238 Second
50MB	180 Seconds	322 Second
60MB	320 Seconds	598 Second
70MB	374 Seconds	721 Second
80MB	427 Seconds	812 Second
90MB	589 Seconds	1093 Second
100MB	650 Seconds	1223 Second



```
    public static String s="abcd";

//static char y[][]= new char[6][6];

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("The Original Text");
        java.io.File file= new java.io.File("SampleTextFile_1000kb.txt");
        try{
            Scanner input= new Scanner(file);

            while(input.hasNext())
            { //int i;
                s= input.nextLine();
               char[] c=s.toCharArray();
            // System.out.println(c);
             //System.out.print(s);
```

```
Input-Integrity (run) ×
The Original Text
Frequency Matrix of given file is as under:
448180      107630      327870      228660      557470      88830
106180      81720       843048      7780        0           600090
384330      484760      384208      178630      109920      483410
499210      498490      760190      132430      0           18960
0           0           0           0           0           0
0           0           0           0           0           0
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figure 3: Screenshot of execution

VI. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

The proposed research introduces an algorithm designed to perform character-level document verification. Initially, a character frequency matrix is created for the document prior to uploading it to the cloud. Upon retrieval, another matrix is constructed, and both are compared. Any mismatch in the entries indicates that one or more characters have been changed, and the algorithm reports the exact altered characters. The algorithm's underlying idea is analogous to traditional error detection and correction methods. At present, it serves as an error detection mechanism by pinpointing altered characters within a document. Future enhancements will focus on developing the capability to automatically restore these characters, thereby achieving complete error correction.

REFERENCES

[1] L. Wei et al., "Security and privacy for storage and computation in cloud computing,"Information Sciences, vol. 258, 2014, pp. 371–386.
[2] https://rb.gy/75beb0
[3] Z. Deng, K. Li, K. Li, and J. Zhou, "A multi-user searchable encryption scheme with keyword authorization in a cloud storage," Future Generation of Computer System., vol. 72, pp. 208–218, 2017.
[4] H. Rasheed, "Data and infrastructure security auditing in cloud computing environments," International Journal of Information Management, vol. 34, no. 3, 2014, pp. 364–368.
[5] L. T. Yang, G. Huang, J. Feng, and L. Xu, "Parallel GNFS algorithm integrated with parallel block Wiedemann algorithm for RSA security in cloud computing," Information. Sciences (NY)., vol. 387, 2017, pp. 254–265.
[6] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," SIAM Journal Computer, vol. 32, no. 3, 2003, pp. 586–615.
[7] A. Azougaghe, Z. Kartit, M. Hedabou, M. Belkasmi, and M. El Marraki, "An efficient algorithm for data security in Cloud storage," 15th Int. Conf. Intelligent System. Design Application,2015, pp. 421–427.
[8] Mohd. Tajammul, RafatParveen and Mohd. Shahnawaz. "Cloud Computing Security Issues and Methods to Resolve: Review", Journal of Basic and Applied Engineering Research, vol. 5, Issue 7; October-December, 2018, pp. 545-550.
[9] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," Journal of Network and Computer. Applications, vol. 34, no. 1, 2011, pp. 1–11.
[10] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," Future Generation of Computer System, vol. 28, no. 3, 2012, pp. 583–592.
[11] M. N. Manas, C. K. Nagalakshmi, and G. Shobha, "Cloud Computing Security Issues And Methods to Overcome," International Journal of Advanced Research in Computer Communication. Eng., vol. 3, no. 4, 2014, pp. 6306–6310.
[12] M. Armbrust et al., " A view of Cloud Computing," Communication of ACM, vol. 53, no. 4, 2010, pp. 50–58.
[13] M. M. Potey, C. A. Dhote, and D. H. Sharma, "Homomorphic Encryption for Security of Cloud Data," Procedia Computer Science, vol. 79, 2016, pp. 175–181.
[14] P. R. Kumar, P. H. Raj, and P. Jelciana, "Exploring Data Security Issues and Solutions in Cloud Computing," Procedia Computer Science, vol. 125, no. 2009, 2018, pp. 691–697.
[15] I. Chuang and S. Li, "An effective privacy protection scheme for cloud computing," ICACT, 13th International Conference,2011, pp. 260–265.
[16] Q. Wang, S. Member, C. Wang, S. Member, and K. Ren, "Enabling Public Auditability and Data Dynamic in Cloud Computing," IEEE Trans. Parallel and Distributed System, vol. 22, no. 5, 2012, pp. 847–859.
[17] M. Tajammul, "Comparative Study of Big Ten Information Security Management System Standards", International Journal of Engineering Research in Computer Science and Engineering, vol. 5, Issue 2, 2018, pp. 5-14.
[18] M. Tajammul, "Comparative Analysis of Big Ten ISMS Standards and Their Effect On CloudComputing",978-1-5386-0627-8/17/$31.00c 2017 IEEE
[19] Tim Mather, SubraKumaraswamy, and S. L, Cloud Privacy and Security. O'REILLY, 2009, pp. 336.
[20] Dan C. Marinescu, Cloud Computing Theory and Practices. Elsevier, 2018, pp. 588
[21] Dijiang Huang and Huijun Wu, Mobile and Cloud Computing, Elsevier, 2018, pp. 336
[22] Naga Malleswari TYJ , Vadivu G , "Adaptive deduplication of virtual machine images using AKKA stream to accelerate live migration process in cloud environment", Journal of Cloud ComputingAdvances, Systems and Applications, 2019, pp. 6-12.https://doi.org/10.1186/s13677-019-0125-z.