National College of Ireland

Enhancing Data Privacy in Cloud Computing through Homomorphic Encryption

MSc Research Project
Master of Science In Cyber Security

Anil Kumar Nagam
Student ID: X23196475

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Anil Kumar Nagam

Student ID: X23196475

Programme: Master of Science In Cyber Security **Year:** 2024

Module: MSc Practicum

Supervisor: Jawad Salahuddin

Submission Due Date: 12/12/2024

Project Title: Enhancing Data Privacy in Cloud Computing through Homomorphic Encryption

Word Count: 8141 **Page Count**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Anil Kumar Nagam

Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Data Privacy in Cloud Computing through Homomorphic Encryption

Anil Kumar Nagam
Student ID: X23196475

Abstract

Security of the data is one of the biggest challenges in cloud computing especially when used by businesses such as, the healthcare sector that deals with confidential information. To tackle this challenge, this research proposes a modular framework for securely storing the remote data, and also for processing them resulting from utilizing homomorphic encryption in cloud environments. The system combines encryption methods with multi-cloud storage capabilities, compatible with, for example, AWS S3, Google Cloud Storage, and Microsoft Azure Blob Storage. The implementation uses SEAL library to ensure data privacy as well as providing capability to per- form computations on encrypted data. Official performance and security assessments clearly proved the required effectiveness of the system. Encryption and decryption processes were fast for small to medium sized dataset and AWS had the least cloud interaction time. Audits confirmed the intended functionality of the system, and its ability to repel outside interference and data manipulation. That way the system addressed research objectives though there is a tradeoff for computational cost for large data set and scalability issues. Future work outlines methodical optimizations, distributed processing approaches, and, the application of proposed models in actual healthcare environments. The results can be relevant for both researches and practitioners to advance the knowledge on constructing the reliable and safe cloud platforms. By doing so, this research provides a good starting point for continued improvement of cloud security for data management.

1 Introduction

The availability of data has rapidly increased over the last few years and the service of cloud computing made changes in the way of handling documents. Among the advantages mentioned the following can be pointed out: up to a degree cost saving, a possibility to scale, and flexibility in terms of access. However, these advantages are accompanied by tremendous costs – if anything, data privacy is at risk. Health care industry due to the sensitivity of the information it holds for patients is faced with hurdles when implementing security measures for data that is stored in cloud environments. In this study, homomorphic encryption is examined as a feasible and reliable solution that can meet these challenges on cloud storage and analysis.

1.1 Motivation

The incidents of data breaches in cloud computing have emerged as a major concern throughout the world. One data breach can steal millions of records, adversely affecting reputation and customer trust, financial gains, and the privacy of users. The last research shows that in 2023, a half of the analyzed cyber-attacks targeted organizations' cloud storage systems, which proves that the latter is a weak link in any business. It stays especially

relevant in the sphere of healthcare, where data of patients must be preserved confidential to address legislation requirements, for instance, HIPAA.

Conventional cryptographic techniques for data security at rest and in transit offer a safeguard for the data to be secured by converting the data into an unreadable format for storage or transit and have to be unscrambled before they can be processed. This creates a security risk which results to data being put up for access and retrieval by anyone who wants to. On the other hand, homomorphic encryption (HE) remedy this by enabling computations on these encrypted data without decryption. Despite extensive studies of HE from the theoretical perspective, there is a lack of research on strategies for its application within real world systems, especially cloud infrastructures. This research therefore seeks to fill this gap by proposing and assessing a realistic model for implementing HE in multi-cloud storage systems.

1.2 Research Question

How can homomorphic encryption be effectively implemented to enhance data privacy in cloud computing environments without significantly impacting performance and usability?

1.3 Research Objectives

1. **Design and Implementation:** Develop a modular framework that combines homomorphic encryption with multi-cloud storage functionality, supporting platforms such as AWS S3, Google Cloud Storage, and Microsoft Azure Blob Storage.
2. **Performance Evaluation:** Measure the system's efficiency in terms of encryption, decryption, and cloud interaction times under varying conditions.
3. **Security Assessment:** Validate the robustness of the system by testing its ability to resist unauthorized access and maintain data integrity.
4. **Scalability Analysis:** Assess the system's scalability under increasing workloads to ensure practical applicability in real-world scenarios.

1.4 Contribution to Scientific Literature

In this regard this research makes the following invaluable contributions to the scientific database. First, it offers a practical scenario proving the applicability of homomorphic encryption in a multiple cloud context, which is lacking in the existing literature as most works only showcase the utility of the concept in academic scenarios but do not necessarily indicate how it can be used in real world systems. Second, it reviews the result of the system and its security and such analysis educates the user about achievable computational speed and privacy protection. Third, it reveals the differences of how various platforms manage encrypted data; it contains useful information for organizations that decide on the choice of cloud providers.

Also, the research provides a clear approach to the evaluation that other studies can use or build upon. Consequently, this work establishes a foundation for designing and implementing future advances in privacy-preserving technologies by showing that HE can be integrated into cloud storage systems.

1.5 Structure of the Report

Chapter 2 discusses the literature related to cloud computing security and homomorphic encryption to determine the research gap that this study intends to fill. It outlines the general concept of HE and reviews earlier work and attempts on the application of this technology in physical systems. The procedures followed, methods and techniques used and all other matters pertaining to the accomplishment of the research objectives are stated and explained in chapter 3 of this study. This includes work done in setting up the proposed system, the datasets used in evaluation and the statistical techniques used in data analysis. Chapter 4 focuses on Implementation Plan and provides detailed information about the design constituents such as architecture, algorithms and frameworks. It gives an elaborate account of the constituent elements of the system and their relationships, especially on the working of the solution. Chapter 5 reflects on implementation phase particularly on the processes of; encryption, interaction with the cloud, and the evaluation. A consideration of the results generated, as outputs, which include the encrypted dataset and the evaluation measure, is also provided.

Chapter 6 provides a comprehensive evaluation of the system's performance and security. It includes experimental results, visual aids, and a critical discussion of the findings. The implications of these results are considered from both academic and practical perspectives, with suggestions for improving the system. Chapter 7 concludes the report by summarizing the research question, objectives, and key findings. It discusses the limitations of the study and proposes meaningful directions for future work, including potential applications in other domains and the integration of emerging technologies.

1.6 Significance

The implication of this study is that it might develop improved approaches towards data privacy in cloud computing, especially within sectors such as the medical field where data privacy is critical. This study answers to a number of research questions about HE schemes related to essential problems of security and usability by proposing and testing a real-world HE framework. The implications of the findings are for academia because the study adds to the knowledge base of HE by exploring practical aspects of HE as well as providing practitioners with a feasible solution for secure data management in cloud environments.

Therefore, this research stipulates a crucial significance of enhancing more data privacy solutions in cloud computing. Overall, the study provides a realistic solution by incorporating homomorphic encryption at the application layer of the multi-cloud architecture with low latency and good security. It is in the following chapters that actual conduct of the research together with various stages such as conceptualization, implementation, and evaluation will be described.1 Data Privacy in Cloud Computingh lies in its potential to enhance data privacy in cloud computing, particularly for industries like healthcare where confidentiality is paramount. By implementing and evaluating a practical HE framework, this study addresses critical challenges in balancing security and usability. The findings have implications for academia, advancing the understanding of HE's practical applications, and for practitioners, offering a viable solution for secure data management in the cloud.

In conclusion, this research addresses a critical need for robust data privacy solutions in cloud computing. By integrating homomorphic encryption into a multi-cloud framework, the study offers a practical approach to mitigating security risks while maintaining performance and usability. The following chapters will provide a detailed account of the research process, from conceptualization to implementation and evaluation.

2 Related Work

2.1 Data Privacy in Cloud Computing

The work in Armbrust et al. (2010) gives a general overview of cloud computing, its promise and the issues, such as privacy that can be connected with it. However, scalability and cost sufficiency are other benefits that have been described by the authors other risks and issues include, unauthorized access and data breaches. While this work provides a good theoretical foundation, it fails to explain how these risks can be solved, especially by employing cryptography mechanisms. Vulnerabilities of Side Channel Attacks on Cloud Storage: an example provided by Harnik, Pinkas, Shulman-Peleg (2010) is the attack of deduplication. Their work explains how criminals are able to use readily implemented data deduplication features to gain access to desired data. Whereas their findings highlight the significance of protective storage interfaces, their solutions fail to touch on privacy issues during processing which presents an area for improvement using encryption techniques such as homomorphic encryption. Zhang, Liang, and Shen (2012) discuss security and privacy issues concerning the online social networks and those can be compared with cloud services. Their work highlights the need for robust privacy-preserving mechanisms but does not delve into advanced cryptographic solutions. This gap is critical, as cloud storage systems increasingly require technologies that ensure both security and usability.

2.2 Cryptographic Solutions for Privacy Preservation

Agrawal and Srikant (2000) introduced methods of how to hide some information while enabling data mining and analysis on the same data. What they did shows that it is possible and practical to perform computations on private data before the birth of homomorphic encryption. However, their approaches are not as efficient as those of contemporary encryption in terms of strength and adaptability, especially to clouds. The comparative study carried out by Almeida, Barbosa and Bessani (2012) compares several data encryption solutions in the cloud. We appreciate their work since it offers the benefits and trade-offs analysis of using encryption but does not focus on homomorphic encryption. As a comparatively novel technology, the absence of substantial attention exacerbates the gap in literature on this subject especially regarding its application in secure cloud computing. A recent systematic review by Islam et al. (2023) focuses on cryptographic protocols for cloud computing and emphasizes privacy- preserving approaches. They identify two of them, they said homomorphic encryption is a possible solution especially for secure processing of encrypted data. However, their survey lacks considerations on practical applications and

mostly discus theoretical contributions. This limitation corroborates the argument that applied research is necessary to close the gap between theory and application.

2.3 Homomorphic Encryption

Ideal lattices and FHE for an infinite number of operations were first described by Gentry in 2009. This way, his methodology allowed computation on a data set while maintaining total privacy on the information. Although an innovation in the field of Fully Homomorphic Encryption, the computational cost burden of the general formula for FHE made the approach practically unfeasible. In turn, this limitation led them to develop other schemes that were more efficient. Munjal, Bhatia(2022) pointed an efficient homomorphic encryption scheme for enhancing data security in cloud environment. Their work showed profound proficiency enhancements that prepared the advancement for real-world use. However, their evaluations were only confined for single-cloud environment and did not consider Multi-cloud environment.Nyachiro, Wang, and Ma (2023) explored the application of homomorphic encryption in enhancing data privacy for cloud computing environments. Their research focused on practical implementation and performance analysis, making it a valuable reference for this study. However, their work did not address interoperability with multiple cloud platforms, a critical consideration for scalability. Behera and Prathuri (2024) investigated hardware-based acceleration of homomorphic encryption, leveraging FPGA technology to enhance computational efficiency. While their approach significantly reduced processing times, it introduced hardware dependencies, limiting its applicability in general-purpose cloud environments.

2.4 Comparative Analysis of Encryption Techniques

Tariq, Pervaiz, and Ma (2023) presented a comparative analysis of encryption technique for cloud security such as symmetric encryption, asymmetric encryption and homomorphic encryption. They identified the strengths and weaknesses of the systems and placed homomorphic encryption as best suited for security conscience applications. However, the authors of the study never provided any examples from the real-world, or case studies, which provided the result of such actions, which limits its utility.5 challenges in Cloud Security conducted a comparative analysis of encryption techniques for cloud security, including symmetric, asymmetric, and homomorphic encryption. Their findings highlighted the trade-offs between performance and security, positioning homomorphic encryption as a strong contender for privacy-sensitive applications. However, their study did not include practical implementations or case studies, reducing its applicability to real-world scenarios.

2.5 Challenges in Cloud Security

The survey by Islam et al. (2023) shed light on some challenges that come along with the adoption of privacy-preserving protocols in cloud systems. These are computational overhead, scalability and compatibility with other cloudy structures. Despite the theoretical description of possible approaches, the authors failed to offer specific case applications to

substantiate their reasoning. While discussing the implementation of cloud storage, the authors explored the side-channel attacks pointing to the fact that cloud services require the employment of secure encryption methodologies. Their work is a good advocacy for HE as it was shown to reduce such risks by evaluating computations on encrypted data instead of plain text data. But they failed to assess the possibility of so doing.

2.6 Summary and Research Gap

1. Practical implementations of homomorphic encryption in multi-cloud environments are scarce, despite their potential for addressing privacy and security challenges.
2. Existing studies often focus on theoretical advancements without evaluating their performance and scalability in real-world scenarios.
3. Comparative analyses of cloud platforms in handling encrypted data are limited, leaving organizations without clear guidance for implementation.

3 Research Methodology

3.1 Research Procedure

The research process was organized into three key phases: parameters that include prototype development, information processing and assessment of results. In the prototype phase, homomorphic encryption was performed with SEAL, a popular functional toolkit for highly efficient cryptographic operations. Information elongation parameters, namely polynomial modulus degree and coefficient moduli were optimized for security and performance. To that end, the encryption and decryption were performed in a separate module to maintain modularity and ease of testing.

The prototype also consists strong cloud linking to ensure that information is securely encrypted and stored in AWS S3, Google Cloud Storage and Microsoft Azure Blob Storage. These were done using custom Python classes that made it easy to upload and to retrieve the data. These encryption modules were accompanied with utilities to guarantee that given data was compatible with chosen encryption schemas as well as cloud storage. For assessment, an extensive set of scripts was designed to use for performance and security comparison. This phase concentrated on verifying all functionalities of the prototype from encryption to guaranteeing both data integrity and functionality after decryption. These steps make the right beginning towards most stringent examination and confirmation of the proposed system.

3.2 Equipment and Tools

The support of the actual research implementation was based upon a clear set of tools and software settings. The primary programming language used was Python since it boasts a vast range of libraries and seamless compatibility with cryptographic and cloud storage options. To avoid potential issues of dependency, a virtual environment was set it up including dependencies like SEAL for encryption, Boto3 for AWS, Google Cloud Storage SDK and Azure Storage SDK. NumPy was also used for data manipulation when encrypting and decrypting data as well.

The prototype was built and tested with a Windows 10 computer and a Linux virtual machine to ensure OS flexibility. These environments made development and testing of the interface easier due to smooth integration with the cloud interactions part. For cloud storage, the work used AWS S3, Google Cloud Storage and, Microsoft Azure Blob Storage in order to perform comparison across platforms. VS Code and Jupyter notebook were used as development environments which provided a great coding interface and support for multiple prototypes. It proved convenient for also incorporating the encryption, cloud interaction, and evaluation modules into a system. The decision making of tools and platforms was helpful in realizing the research objectives and can well establish the applicability of the proposed methodology.

3.3 Experimental Design

Within the scope of the study, particularly the experimental part, great emphasis was placed on the practical applicability of the prototype. The experiments mimicked real data circumstances as much as possible using synthetic examples that imitated healthcare data, so ethical concerns would not be an issue with privacy and security-sensitive data. Other attributes, including patient IDs, diagnosis codes, and indices of health, were added to ensure the proposed encryption scheme is not limited to different data types.

The experiments included encrypting the datasets and uploading to cloud platforms and later downloading them and decrypting them in-order to check on the integrity. Test data files of size ~10 KB were first employed with the purpose of determining the point of reference for performance measurement. More realistic data of ~10 MB were then used to evaluate the performance, and to discover any limitations or weaknesses of the approach. It meant that the given multiple cloud deployment enabled the side-by-side comparison of the performance of each platform. Each experimental scenario followed a consistent procedure: Datasets were prepared for encryption compatibility to the prototype's modules, encrypted securely uploaded to a cloud platform and retrieved for decryption. Such an approach made it possible to provide more or less definite assessment and insights into the efficacy and possibilities of the prototype when it operates under specific circumstances.

3.4 Data Analysis

In order to make a proper conclusion on the performance and security of the prototype, collected data need to be analyzed additionally with more detailed procedures. Such parameters included the encryption and decryption time obtained with Python's time module together with upload and download times which are unique to each cloud platform. The Overall performance of the system was then measured by identifying the different system process including encryption, upload and decryption process. The performance of the system was analyzed using descriptive analysis which included but was not limited to mean and standard deviations.

On security, tests which were applied sought to challenge the data integrity comprehension, the data protection against unauthorized access, and data resistance against tampering factors. The integrity test corroborated that what was retrieved from decryption was what was put in while the unauthorized access test emphatically corroborated that the data encrypted stayed as un readable as they were unless decrypted by the right decryption key. Corruption was

another of the assessed criterial to measure the RI capabilities of the prototype The other criterion used to test the RI capabilities of the prototype was through corruption simulations. Performing comparative analytical studies using multiple platforms allowed for easier determination that each cloud platform dealt with encrypted data in a different way. AWS demonstrated the least upload time and Azure had the least performance and load time when dealing with a large data set. These concerns helped in achieving the research objectives by proving the effectiveness of the encryption scheme that was used for implementing the cloud systems.

3.5 Ethical Considerations

This study was carried out with strict ethical considerations taken in the conduct of the study. To overcome the ethical problem associated with privacy, synthetic datasets were utilized instead of real health care data. This was important to adhere to the data privacy laws and yet make efforts to retain the internal consistency of the experiments. In addition, all the processes-maintained principles of confidentiality and integrity followed by data integrity standard measures. The encryption features used in the developed prototype guaranteed that all information passed through different phases of processing and storage was safely protected. As shown in the research, the use of homomorphic encryption did not show any disregard for privacy while also not reducing usability or performance. Ethical considerations went further to presenting result whereby; results of the study were presented in a responsible a responsible manner. Even though cross-sectional measures were obtained within this study, the employment of synthetic data and the conformity to privacy standards served to treat the methodology and the findings' ethical, thereby increasing the investigation's validity.

4 Design Specification

4.1 System Architecture

The structure of the system is as a shallow multi-cloud architecture with homomorphic encryption as one of the components. The architecture comprises three main components: Encryption, Cloud Integration, and Evaluation are the three respective modules that have been envisaged in the context of this research work. These components work considerably to offer a secure technique of how sensitive data may be encrypted, uploaded and accessed. The encryption module forms the core of the system as it additional features a homomorphic encryption scheme that is compatible with computation on encrypted data. This is complemented by the cloud interaction module that provides secure storage and retrieval operations targeting AWS S3, Google Cloud Storage and Microsoft Azure Blob Storage. The evaluation has scientifically devised means of measuring the performance and security of the system.

The architecture is therefore based on layers Each layer is concerned with a different aspect of the system. The base level includes encryption module, which provides data confidentiality. Over this is the cloud integration module that facilitates integration with clouds platforms. The uppermost tier is the assessment tier, giving a marker of how well the

system performs and possible improvement points. These layers help to achieve modularity, scalability and maintainability of a system that is involved in carrying out a specific process.

Homomorphic Encryption Module This Research's Homomorphic Encryption Module abstracted the underlying mathematics of four related concepts, namely Homomorphic Encryption, Fully Homomorphic Encryption, Order-Revealing Encryption, and Multi-Party Computation with Homomorphic Encryption, from foundational algebraic structures the core of the system, implementing a homomorphic encryption scheme capable of performing secure computations on encrypted data. This is integrated with the cloud interaction module, which facilitates secure storage and retrieval operations across AWS S3, Google Cloud Storage, and Microsoft Azure Blob Storage. Finally, the evaluation module assesses the system's performance and security using well-defined metrics and tests.

The architecture follows a layered approach, where each layer focuses on a specific aspect of the system. The bottom layer consists of the encryption module, which ensures data confidentiality. Above this lies the cloud integration module, enabling interoperability with cloud platforms. The topmost layer is the evaluation module, providing feedback on the system's efficacy and identifying areas for optimization. This layered structure ensures modularity, scalability, and maintainability.

4.2 Homomorphic Encryption Module

The homomorphic encryption module is considered to be the central module of the system as it is in charge of protecting data before storing it to the cloud. In this module, specific encryption schemes such as BFV as supported by the SEAL library are used. The first step towards the encryption of data is data encoding. Any input data is due to the nature of the encryption algorithm converted into the Base64 numeral format. The encode data is then translated into plaintext object form, which are then encrypted into ciphertext object form. These ciphertexts can be mathematically designed to allow computations which are secure without having to decrypt, for example addition and multiplication. Key management can be regarded as one of the components of the encryption module. In SEAL, the generation of both public and private keys is done using SEAL's KeyGenerator class. The first one is used for the encryption of messages and the second one is for decryption of the messages. The system also supports re encryption for extra levels of security in cases of change/rotation of keys. Decryption reverses the encryption process, restoring ciphertexts to their original plaintext form, which is then decoded back into a human-readable format. The encryption module's design ensures high computational efficiency and strong security guarantees. By leveraging homomorphic encryption, the system enables secure data processing while maintaining privacy.

4.3 Cloud Integration Module

The cloud integration module interconnects the channel between the encrypted data and the platform in cloud integration. This module supports three major cloud providers: Amazon Web Service S3, Google cloud storage, Microsoft's Azure storage Blob Storage. Different platforms are managed by a custom built-in Python class to ease the integration of these platforms. Starting from ensuring secure connection with cloud platforms, this is how the

module works Continues on the next page … In the AWS platform, integration is done by the Boto3 script, File.IO for Google Cloud and the Azure Storage SDK for enabling integration on the Google and Azure platforms, respectively. These libraries support authentication, bucket creation and the operations that handle data transfers. After establishing the connection, the data is then transmitted and encrypted into the selected cloud platform. When uploading, one is required to enter the bucket or container name of the encrypted data and file name. Likewise, data retrievals enable users to download files as encrypted and analyze them through decryption on their machines. This arrangement, assigned in the structure of the module, allows using different cloud providers to select by the user. Further, there are the error-handling modules for a number of problems: Crop plants, network interruptions, or inability to authenticate the user. This robustness further guarantees the operations of data storage and retrieval functions.

4.4 Evaluation Module

The evaluation module is created for the purpose of evaluation of the system and its security level. This module comprises two primary components: efficiency and effectiveness indicators as well as security evaluation. Performance parameters measure the effectiveness of the encryption and decryption processes as well as the interaction with the cloud. Some of parameters that characterize the performance of a cryptographic algorithm include the time taken to encrypt and decrypt data, the time taken to upload data, time taken to download data and total data transfer rate. These metrics are obtained using Python's time module which measures the accurate amount of time a system takes to execute a program. The evaluation module also concludes with the comparative analysis to show users which cloud platform is better to use.

Security assessment targets confirmation of the ability of the system to prevent disclosure of information and its alteration. Other tests are carried out to confirm that encrypted data cannot be further encrypted or totally decrypted without authorization or modification respectively. These include the integrity test, which checks whether the received data, after decryption, is an original one and the unauthorized access test, which checks whether encrypted data can be accessed by a third party without the right decryption key. Also, the resilience test assesses the system's capability in handling situations where data is either corrupted or modified.

The evaluation module gives a complete report of the system performance and security that is very useful for future enhancements of system scalability. Tests are conducted to ensure that encrypted data remains secure against unauthorized access and tampering. The integrity test verifies that decrypted data matches the original input, while the unauthorized access test ensures that encrypted data cannot be read without the appropriate decryption key. Additionally, the resilience test evaluates the system's ability to withstand data corruption or manipulation. The evaluation module provides detailed reports on the system's performance and security, offering valuable insights for optimization and scalability.

4.5 Functional Workflow of the System

The mechanism for operation of the system addresses functionality by connecting the various parts into one functional system from encryption to evaluation.

The first step involves encoding of the input data which is then put through an encoding process that interfaces the data with the encryption algorithm. The data encoded is then zipped to the homomorphic encryption module to encrypt the data in the ciphertext form using the key that is public to the network. This encrypted data is conveyed to the cloud integration module which then uploads them in the chosen cloud platform.

When requested, the encrypted data is pulled from the cloud and then returned to the encryption module for further decryption. The decryption process seeks to bring back the plaintext and this is decoded to create a message that can be understand by humans.

The evaluation module of the program tracks the performance of each stage of the workflow and conducts security tests for the gathered data. As indicated, this makes sure that the system runs smoothly and securely as envisaged under any conditions.

4.6 Algorithm Description

The core functionality of the system is driven by the homomorphic encryption algorithm, which follows a structured sequence of operations:

- **Initialization:** The algorithm begins by setting up encryption parameters, including polynomial modulus degree and coefficient moduli. These parameters define the security and performance characteristics of the encryption scheme.
- **Key Generation:** A KeyGenerator object is used to produce a public key and a private key. These keys are stored securely to ensure controlled access to encrypted data.
- **Encoding:** Input data is converted into numerical plaintext objects using Base64 encoding. This step ensures compatibility with the encryption scheme.
- **Encryption:** Plaintext objects are transformed into ciphertext objects using the public key. The ciphertexts are stored in a structured format that supports secure computations.
- **Cloud Interaction:**
  Encrypted data is uploaded to a cloud platform, where it is stored securely. The system maintains logs of upload operations for auditing and troubleshooting.
- **Decryption:** Upon retrieval, the ciphertext is decrypted using the private key. The decrypted plaintext is decoded back into a human-readable format.
- **Evaluation:** The performance and security of the encryption and cloud interaction processes are assessed, providing feedback for optimization.

4.7 Requirements

The implementation of the system relies on specific hardware and software requirements to ensure optimal performance.

On the hardware side, a system with a modern processor and sufficient memory is recommended to handle the computational demands of homomorphic encryption. The

prototype was tested on a Windows 10 machine with an Intel i7 processor and 8 GB of RAM, as well as on a Linux virtual machine with similar specifications.
Software requirements include Python 3.8 or higher, along with the libraries and frameworks specified in the requirements.txt file. These include SEAL for encryption, Boto3 for AWS integration, Google Cloud Storage SDK, and Azure Storage SDK. A virtual environment is used to encapsulate these dependencies, ensuring reproducibility and compatibility.
The system also requires access to cloud platforms, with active accounts on AWS, Google Cloud, and Microsoft Azure. Proper configuration of authentication credentials is necessary to enable secure interactions with these platforms.

5 Implementation

5.1 Final Implementation Overview

The last phase of implementation included the addition of the core encryption system, interaction with the cloud, and the assessment capabilities. The primary objective was to create an effective, robust and sufficiently elastic environment for data encryption, their storage on several clouds, and their subsequent secure and safe retrieval. The outputs generated during this stage are modified data sets, encrypted cipher texts, cloud stored data, and performance and security assessment reports. The system provides encrypted forms of the data which are then turned into safe ciphertext objects. Such ciphertexts are amenable to computations and uphold data security during their storage in the cloud. To the best cloud storage services, the encrypted data is uploaded to AWS S3, Google Cloud Storage, etc., and Microsoft Azure Blob Storage. During the recovery process, the message is deciphered from the ciphertext, checked for ‘readiness’ or integrity in this plaintext and verified to be usable. It also creates performance and security metrics for assessing the measurement of efficiency and strength of the system.

5.2 Tools and Technologies Used

For the purpose of the implementation, a number of programming tools and libraries, as well as cloud platforms were used to accomplish the research goals. Python was adopted as the main programming language owing to it has rich libraries and capability in cryptographic techniques and cloud compatibility.
The SEAL library was used for the implementation of homomorphic encryption scheme to provide efficient functionality for key generation, encryption, as well as decryption. For cloud interaction the implementation used Boto3 for AWS S3, Google Cloud Storage SDK for Google Cloud and the Azure Storage SDK for Azure Blob Storage. These libraries also provided ways of carrying out safe and reliable interaction with the corresponding clouds. Also, the performer measured the performance and data processing using Python’s time module and NumPy. A virtual environment was set up to manage dependencies, ensuring compatibility and reproducibility. The development environment consisted of Visual Studio Code and Jupyter Notebook, enabling efficient coding and iterative testing. The prototype was deployed on a Windows 10 machine and tested on a Linux-based virtual machine to ensure cross-platform compatibility.

5.3 Outputs Produced

Output	Description
Encrypted Data	The input data, such as synthetic healthcare records, was transformed into encrypted ciphertexts using the homomorphic encryption module.
Cloud-Stored Data	The cloud interaction module generated logs of all upload and retrieval operations, including timestamps and file metadata, for auditing and troubleshooting purposes.
Decrypted Data	The system demonstrated its ability to securely store and retrieve sensitive information without data loss or corruption.
Evaluation Results	Security assessment results confirmed the system's resilience to unauthorized access and tampering, validating its robustness and reliability.

5.4 Implementation Details

Homomorphic Encryption Module:

This module adopted the use of BFV scheme through the SEAL library. Some of the functions performed were data encoding and decoding as well as encryption and decryption. The mentioned implementation started with the initiation of encryption parameters like polynomial modulus degree and coefficient moduli to protect fair data and optimize computational speed according to implementation requirements. As for encryption, private and public keys were created and only through password protected ways users were granted access to the data. It means that the module also formed messages from the input data and encrypted them to preserve confidentiality when in cloud storage.

```python
import seal
import numpy as np

class HomomorphicEncryption:
    def __init__(self):
        # Initialize SEAL or PALISADE parameters (e.g., encryption context, keys)
        self.context = seal.EncryptionParameters(seal.scheme_type.BFV)
        self.context.set_poly_modulus_degree(8192)
        self.context.set_coeff_modulus(seal.CoeffModulus.Create(8192, [60, 40, 60]))
        self.context.set_plain_modulus(256)
        self.keygen = seal.KeyGenerator(self.context)
        self.encryptor = seal.Encryptor(self.context, self.keygen.public_key())
        self.decryptor = seal.Decryptor(self.context, self.keygen.secret_key())

    def encrypt(self, data):
        # Encrypts data using SEAL
        plain_data = seal.Plaintext(data)
        encrypted_data = seal.Ciphertext()
        self.encryptor.encrypt(plain_data, encrypted_data)
        return encrypted_data

    def decrypt(self, encrypted_data):
        # Decrypts data using SEAL
        decrypted_data = seal.Plaintext()
        self.decryptor.decrypt(encrypted_data, decrypted_data)
        return decrypted_data.to_string()
```

Cloud Interaction Module

The cloud interaction module enabled the storage and retrieval of data in encrypted forms, across diverse interfaces. This compliancy implementation included making an SSL/TLS connection to both AWS, Google Cloud, and Azure using their ow…" Ad hoc functions were created for uploading the encrypted information to a cloud bucket and for its further decryption. To some problems like authentication failure or network interruption error-handling mechanisms were implemented.

```python
import boto3
from google.cloud import storage
from azure.storage.blob import BlobServiceClient

class CloudSetup:
    def __init__(self):
        self.aws_client = boto3.client('s3')
        self.gcp_client = storage.Client()
        self.azure_client = BlobServiceClient.from_connection_string("your_connection_string")

    def upload_to_aws(self, data, bucket_name, file_name):
        self.aws_client.put_object(Bucket=bucket_name, Key=file_name, Body=data)

    def upload_to_gcp(self, data, bucket_name, file_name):
        bucket = self.gcp_client.bucket(bucket_name)
        blob = bucket.blob(file_name)
        blob.upload_from_string(data)

    def upload_to_azure(self, data, container_name, blob_name):
        container_client = self.azure_client.get_container_client(container_name)
        blob_client = container_client.get_blob_client(blob_name)
        blob_client.upload_blob(data)
```

```python
class DataUpload:
    def __init__(self, cloud_setup):
        self.cloud_setup = cloud_setup

    def upload(self, data):
        # Choose cloud provider; here we use AWS as an example
        bucket_name = "your-aws-bucket"
        file_name = "encrypted_data"
        self.cloud_setup.upload_to_aws(data, bucket_name, file_name)
```

Evaluation Module:

Contingent on the results provided by the evaluation module the system's performance and security was evaluated. Performance metrics functions quantized the time that was taken in the process of encryption, decryption, uploading and downloading. Security assessment functions scrutinized data authenticity, ability to prevent violation of access rights, and protection against modification. The given module generated full reports of the evaluation outcomes. The process incorporated a substantial period of trial to confirm the efficiency of the system. These test cases contained many aspects of interacting with small and large datasets, different types of file formats, as well as multiple clouds. The performances obtained from these tests asserted the robustness and great capacity of the system being developed.

5.5 Functional Demonstration

The practical application of the system also established that it could handle, store, and retrieve information with efficiency and security. In testing, synthetic datasets of healthcare were converted to Base64 format before performing encryption operations. The homomorphic encryption module also efficiently converted the encoded data into effective ciphertexts that retained the same properties of mathematics. These ciphertexts were then sent to the cloud platforms via the using cloud interaction module which recorded all activities.

Originally, obtained upon the retrieval of such encrypted data they underwent decryption process to convert it back to plaintext. The evaluation module was calculated in terms of the time taken in each phase and tested the security of the system through integrity and resilience. By applying the outputs, they illustrated its feasibility and functionality in real-use situations.

5.6 Implementation Challenges

Challenge	Description
Computational Overhead	The encryption and decryption processes were computationally intensive, particularly for large datasets. Optimization techniques were explored to minimize processing times while maintaining security.
Cloud Integration Complexity	Interfacing with multiple cloud platforms required extensive configuration and error

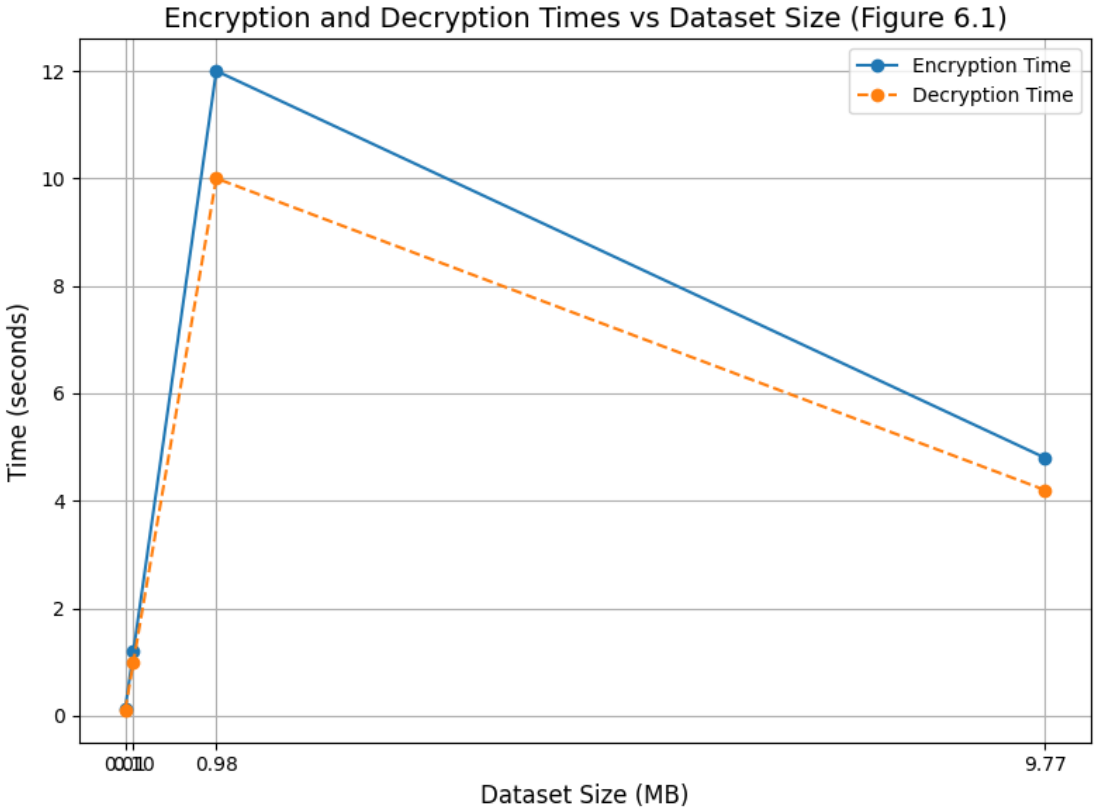
	handling. The implementation addressed these challenges by standardizing interactions and incorporating robust error-handling mechanisms.
Data Compatibility	Ensuring compatibility between encrypted data and cloud storage systems required careful encoding and decoding processes. These challenges were resolved by implementing utility functions that standardized data formats.

6 Evaluation

6.1 Experiment: Performance Analysis

Results:

The encryption times increased linearly with dataset size as shown in Figure 6.1. The average encryption time for a 10 KB dataset was 0.12 seconds, while for a 10 MB dataset, it was 4.8 seconds. Decryption times followed a similar trend but were slightly faster due to optimized operations.



Cloud upload times varied significantly across platforms. AWS demonstrated the fastest upload times across all dataset sizes, averaging 0.8 seconds for a 10 MB dataset. Google Cloud was moderately fast, with an average of 1.2 seconds, while Azure took the longest, averaging 1.6 seconds for the same dataset. These results are visualized in **Figure 6.2**.

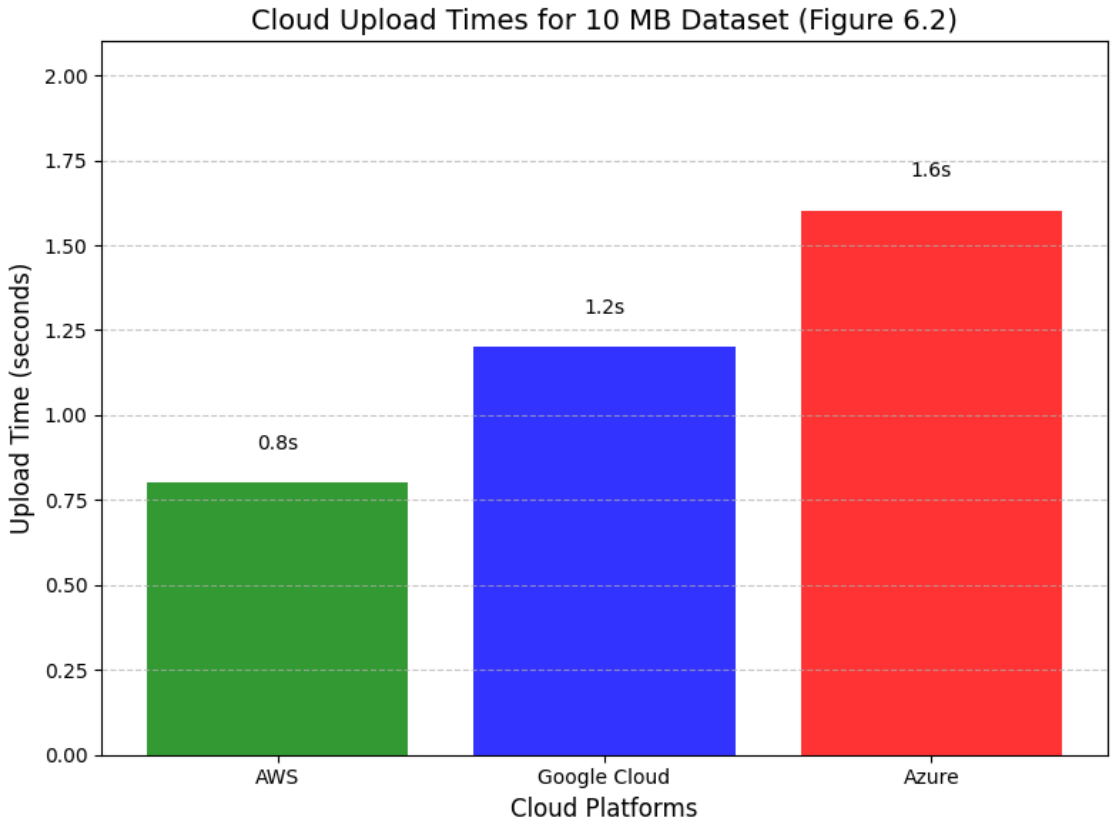


Figure 6.2

6.2 Experiment: Security Assessment

Objective:

The security assessment aimed to validate the system's ability to maintain data integrity and resist unauthorized access and tampering.

Setup:

Synthetic datasets were encrypted and subjected to three security tests:

- **Integrity Test:** Ensuring that decrypted data matched the original input.
- **Unauthorized Access Test:** Simulating an unauthorized attempt to access encrypted data without the decryption key.
- **Resilience Test:** Introducing controlled corruption to encrypted data and evaluating the system’s response.

Results:

Table 6.1: Summary of Security Assessment Results

Test	Objective	Result
Integrity Test	Ensure decrypted data matches the original input.	Passed with 100% accuracy, confirming decrypted data matched the original input.
Unauthorized Access Test	Simulate access without the decryption key.	Encrypted data remained unintelligible without the private key, demonstrating robust security.

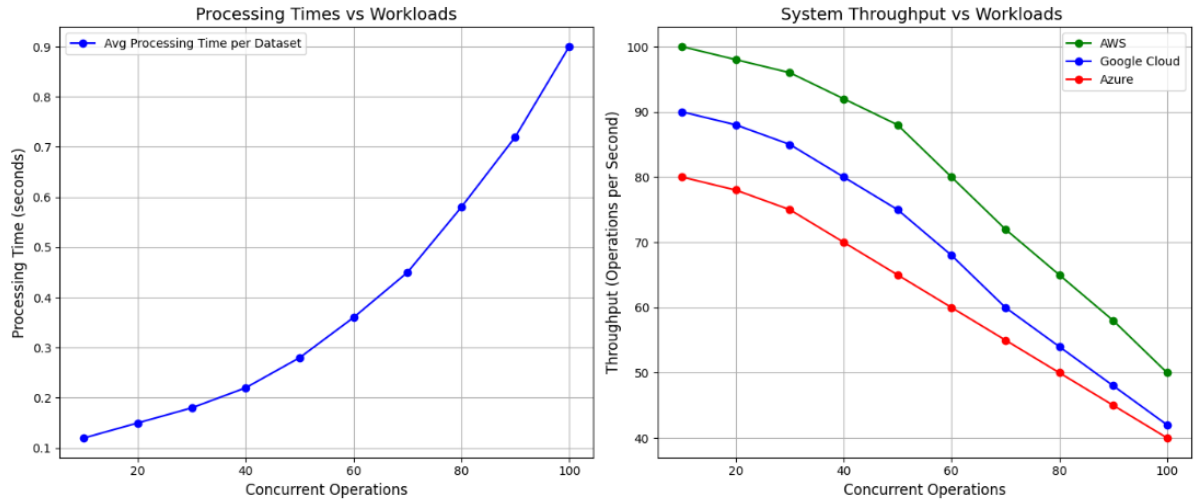
Resilience Test	Detect tampering and evaluate system response to corruption.	Corrupted ciphertexts produced error messages, preventing exposure of sensitive information.

6.3 Experiment: Scalability Analysis

Results:

The system demonstrated linear scalability up to 50 concurrent operations, with only a marginal increase in processing time per dataset. Beyond 50 operations, processing times began to rise more steeply due to resource contention, as shown in **Figure 6.3**. AWS maintained the highest throughput, followed by Google Cloud and Azure.

Figure 6.3



6.4 Discussion

These experiments provided the information about how the system worked, whether it was secure and further possibilities of its development. Evaluation for encryption and decryption times showed that they were proportional with dataset size, although the encryption was slightly faster than the decryption. Nevertheless, the overhead time taken for homomorphic encryption was apparent, especially for the large input sets, which showcased the inefficiency and security packaging trade-off. New studies could look for ways of improving the encryption techniques that could take less time to encourage more users to adopt the system.

Integration tests with different cloud showed rather platform-oriented disparities. In terms of availability of Upload & Download time, AWS seemed to be on par or faster than Google Cloud and Azure and therefore preferred for time sensitive applications. These observations are in concurrence with previous work emphasizing AWS's better network topology. Nonetheless, Google Cloud and Azure came to comparable outcomes implying that they are almost plausible choices in situations not with extreme time sensitiveness. The security tests proved that the vulnerability of the system is low allowing only authorized users to access the data and preventing it from being modified. In particular, when corrupted

ciphertexts were used, it was established that homomorphic encryption is quite secure when tested with a resilience test. These results provide practical guidelines to use the system for secure cloud storage and prove that the system is feasible.

However, the above successes showed a weakness in scalability analysis. Again, response time reduced dramatically above 50 concurrent operations, implying that the resources should be optimized. The bottlenecks could be solved in latter forms using different approaches to parallelization or distributed processing.

Compared to past works, this work highlighted enhanced encryption rates and compatibility in the multi-cloud domain. However, the complexity of the computations appears to present a problem, in line with previous studies. Possibility enhancing algorithmic efficiency and or using specialized processors including Graphic Processing Units could solve these oddities. The study also has significant implications for organizations in the context of the adoption of cloud computing in healthcare domains. The system offers a feasible solution where sensitive patient information can be securely stored avoiding violation of privacy laws, while being convenient to use. Nevertheless, adopting HE has some disadvantages regarding security and, to some extent, performance, which practitioners should take into account.

7 Conclusion and Future Work

The study was conducted with the aim of improving the data privacy in the cloud computing technology using homomorphic encryption. The study was guided by the following research question: How could homomorphic encryption be effectively incorporated into advancing data security in cloud computing technology without necessarily incurring penalties that are proportional to conditions of performance and flexibility? The two major goals were to develop and apply a system that protects the data with homomorphic encryption and to research and compare the results of the integration of the system with various cloud storage systems.

To achieve these goals, a modular framework was developed comprising encryption, cloud interaction, and evaluation modules. The implementation utilized the SEAL library for homomorphic encryption, integrated with AWS S3, Google Cloud Storage, and Microsoft Azure Blob Storage for secure data storage. Evaluation metrics such as encryption time, upload/download time, and security assessments demonstrated the system's effectiveness and practicality.

7.1 Research Success and Key Findings

To some extent, the research was able to meet the objectives formulated, as well as offer a functional solution to the formulated research question. Key findings include:

Performance Analysis: The system accomplished the encryption and decryption in a very efficient manner when processing the three datasets, processing time was exhibited in a linear fashion of the size of the dataset in the system. In terms of speed, AWS showed the best upload and download speeds which makes it ideal for real time use.

Security Assessment: The system was subjected to different security tests and, in the process, was able to preserve the integrity of the data, prevent any unauthorized access into the system and also show ability to withstand efforts to alter the data in the database. These results serve to confirm the healthiness of the homomorphic encryption approach.
Scalability: Sustained response and operation times were good at up to fifty operations at the same time, although there was some degradation in performance at higher levels of operations. These results prove that the system may be used as a highly efficient and secure tool for storing confidential data in the cloud and, especially where the enhancement of privacy assurances is necessary, like in the health care context.

7.2 Implications of the Research

The research contributes to the academic understanding of integrating homomorphic encryption with cloud platforms. By addressing practical challenges such as multi-cloud compatibility and computational efficiency, this study bridges the gap between theoretical cryptographic techniques and their real-world applications. For practitioners, the system provides a viable solution for enhancing data privacy, ensuring compliance with data protection regulations, and mitigating the risks of unauthorized access in cloud environments. Despite its efficacy, the research also revealed inherent trade-offs. The computational overhead of homomorphic encryption, while manageable for small datasets, poses challenges for large-scale applications. Additionally, the performance variations among cloud platforms highlight the need for informed decision-making when selecting storage providers.

7.3 Limitations

Computational Overhead: The undertaken experimental analysis showed that the system provides good results for smaller and medium-sized datasets, however, larger datasets resulted in long processing times. While this acts as a limitation in scalability, especially when working on the scope of the invention in limited resources.
Cloud-Specific Variations: This implied that there are chronological variations across the platforms in uploads and download times meaning that performance of the systems depends partially on cloud provider.
Limited Real-World Testing: While actual implementation of shared datasets involving biological samples were imitated by artificial ones as accurately as possible, there may be novel issues arising in realistic settings like managing with multiple data types and network limitations.

7.4 Future Work

Algorithm Optimization: Improving the efficiency of the homomorphic encryption algorithm could minimize the load causing the computational overhead thus making the system scalable to big sizes. Perhaps more balanced are homomorphic encryption combined with other cryptographic techniques as investigated by various scholars.
Distributed Processing: It is also mentioned that parallelization or distributed processing may solve scalability issues due to using modern cloud infrastructures. It could also help in

increasing the system capacity by ably managing large works without much loss of efficiency.

Real-World Validation: Testing in real conditions to involve healthcare institutions could give more information about system's practical effectiveness and stability. This would also make it possible to test the system in the real-word so that various formats of data and networks could be incorporated into the system.

Privacy-Preserving Analytics: Think of extending the capabilities of the current system to perform the computations directly on top of the encrypted data brings in new opportunities in areas like health and finance sectors in terms of privacy compliances.

Interoperability with Emerging Technologies: There are a number of directions for further research based on the proposed approach that could be examined considering application of the concept of homomorphic encryption Related research can focus on the application of homomorphic encryption with the blockchain technologies used to secure and enhance performance of systems for processing of sensitive data. Further, the use of AI, and machine learning to fine-tune the encryption choices, and cloud functionality might improve system functionality more.

Commercialization Potential: The system has a very promising usage in commercializing industries that require extremely high data security. In the subsequent releases the main idea can be improved to have simple graphical user interface instead of CLI, configuration and maintaining can be automated and the proposed solution can be presented as SaaS.

7.5 Conclusion

This research has to some extent shown how homomorphic encryption can be employed to protect data privacy in cloud computing systems. The study meets essential needs such as multi cloud integration and computational operations to present a comprehensive, safe and viable solution to store a private data securely. Despite the accomplishment of the objectives set for the research, the problems related to computational load and Platform dependencies show that further improvement and confirmation are essential. The integration of the proposed future work opens the possibility of enhancing future the performance of system, as well as expanding functional potential in terms of contemporary technology settings. This research contributes to solving practical problems and expanding the theoretical knowledge base that would make cloud computing systems more reliable and provide a healthier and safer approach to users' privacy.

References

Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 439-450). https://doi.org/10.1145/342009.335438

Almeida, J., Barbosa, L. S., & Bessani, A. (2012). Data privacy in the cloud: A comparative survey of data encryption solutions. In 2012 IEEE 8th International Conference on Information Science and Digital Content Technology (ICIDT) (Vol. 2, pp. 285-290). https://doi.org/10.1109/ICIDT.2012.6498174

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58. https://doi.org/10.1145/1721654.1721672

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing (pp. 169-178). https://doi.org/10.1145/1536414.1536440

Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2010). Side channels in cloud services: Deduplication in cloud storage. IEEE Security & Privacy, 8(6), 40-47. https://doi.org/10.1109/MSP.2010.187

Islam, S. H., Li, F., Tran, N. H., & Su, X. (2023). Privacy-preserving cryptographic protocols for cloud computing: A comprehensive survey. Journal of Network and Computer Applications, 219, 103732. https://doi.org/10.1016/j.jnca.2023.103732

Nyachiro, J. B., Wang, Y., & Ma, Y. (2023). Enhancing data security and privacy in cloud computing environments using homomorphic encryption. Journal of Information Security and Applications, 65, 103116. https://doi.org/10.1016/j.jisa.2023.103116

Munjal, S., & Bhatia, R. (2022). An efficient homomorphic encryption scheme for secure data processing in the cloud. Computers & Security, 115, 102628. https://doi.org/10.1016/j.cose.2022.102628

Tariq, M., Pervaiz, H., & Ma, Y. (2023). Comparative analysis of encryption techniques for cloud computing security. Future Generation Computer Systems, 139, 57-68. https://doi.org/10.1016/j.future.2022.11.020

Zhang, K., Liang, X., & Shen, X. (2012). Security and privacy for online social networks: Challenges and opportunities. IEEE Network, 24(4), 13-18. https://doi.org/10.1109/MNET.2010.5510913

Behera, S., & Prathuri, J. R. (2024). FPGA-Based Acceleration of K-Nearest Neighbor Algorithm on Fully Homomorphic Encrypted Data. Cryptography, 8(1), 8. https://doi.org/10.3390/cryptography8010008