National College of Ireland

Enhancing Cloud Data Security and Storage: Integrating Zero-Knowledge Proofs with Lightweight Homomorphic Encryption for Efficient Deduplication

MSc Research Project
Cloud Computing

Sunandan Sekhar Das
Student ID: 23135417

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sunandan Sekhar Das
Student ID:	23135417
Programme:	MSc in Cloud Computing **Year:** 2023-2024
Module:	MSc Cloud Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	12-08-2024
Project Title:	Enhancing Cloud Data Security and Storage: Integrating Zero-Knowledge Proofs with Lightweight Homomorphic Encryption for Efficient Deduplication
Word Count:	**9374 Page Count** 23

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: Sunandan Sekhar Das

Date: 12-08-2024

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 Cloud Data Security and Storage: Integrating Zero-Knowledge Proofs with Lightweight Homomorphic Encryption for efficient Deduplication

Sunandan Sekhar Das
23135417

Abstract

The exponential growth of cloud computing has necessitated advanced solutions for secure and efficient data management. This research presents a novel framework that combines Lightweight Fully Homomorphic Encryption (LTFHE), Zero-Knowledge Proofs (ZKP), and deduplication techniques to address the challenges of data security, integrity, and storage efficiency in cloud environments. Based on the SEAL (Simple Encrypted Arithmetic Library) for performing arithmetic operations on encrypted data, the proposed framework facilitates secure computations without compromising privacy. Hence, incorporating the Schnorr-based ZKP protocol guarantees secure data integrity check without compromising the privacy of the user's data in real-time application. Based on a set of experiments carried out on the Amazon EC2 instances, the framework was benchmarked for efficient performance through file sizes of 1 MB up to 200 MB. The performance analysis substantiates linearity of both encryption and decryption algorithms in relation to the file size however, additional enhancements are required to improve the decryption of bigger file sizes. The ZKP protocol had fairly negligible overhead and at the same time, was highly reliable which proves that it is effective in securing cloud storage. Furthermore, the deduplication procedure also worked effectively in pointing out the duplication of data to ensure reduced storage and overall system efficiency. By using advanced cryptographic techniques this research provides a scalable and practical solution to the cloud security problems. The results presented here lend further support to the practical applicability of the proposed framework for future research and for related technologies in commercial cloud-based storage. Further work will involve fine-tuning the decryption process and expanding on the ways of applying these techniques in large-scale cloud infrastructures.

1 Introduction

The adoption of cloud storage solutions has significantly transformed data management by enabling scalable and flexible storage capabilities. However, the exponential increase in data volume has also posted significant problems in matters concerning security as well as efficiency in these systems. As the data is stored and processed in the cloud environments, the risks of data breaches, unauthorized access, and privacy violations are higher, and therefore, proper security measures should be incorporated. On the same note, the need to find ways of dealing with redundancy and how to effectively store items in the limited space available has been more pronounced than ever.

Over the past few years, the market for cloud storage has been rapidly expanding due to the large amounts of data being produced by corporations and customers. According to Marketsandmarkets, the global cloud computing market is expected to nearly double from $626.4 billion to $1,266.4 billion by 2028, with a compound annual growth rate (CAGR) of 15.1% (Marketsandmarkets, 2023). Likewise, Gartner (2023) reveals that organizations' expenditure on public cloud services is expected to grow to $597. 3 billion, which indicates the need for more cost-efficient solutions delivered through cloud environments. However, there are still issues of security and 39% of companies have known data breaches in the cloud environment in 2022 according to Thales Group (2023). This is why there has been a call for enhanced security measures especially when it comes to data deduplication.

Even though data deduplication is a critical process for eliminating the redundant data and thus, enhancing the efficiency of storage, it has its specific issues when implemented in cloud settings. Conventional methods of deduplication have been known to pose a threat to the privacy of the data because some of the data has to be viewed to identify the duplicates. This poses a challenge between the need to have storage space and the need to have high security on data. The proposed solution is based on the Lightweight Fully Homomorphic Encryption (LTFHE) and Zero-Knowledge Proofs (ZKP). ZKP can make sure that the data is unique, and LTFHE can perform computations on encrypted data so that the data remains private at all times during the deduplication process.

Motivated by this significant research problem, this study poses the following open-ended research question: **How can the integration of Zero-Knowledge Proof (ZKP) with Lightweight Fully Homomorphic Encryption Over the Torus (TFHE) ennhance data deduplication in cloud storage for better privacy and efficiency of data storage**? This paper discusses these approaches to guarantee the privacy and security of the data during the deduplication process as a way of solving the main issues that were highlighted in the literature review section.

2 Related Work

In the digital world, the era of digital breakthrough is facing a big challenge in the increasing capacity of storage in the cloud while maintaining the security of data storage. Cloud storage is the ideal contemporary approaches to data management that employ methods that address the need for data confidentiality and integrity, as well as storage needs. In this section, related work of data deduplication is discussed, which can undoubtedly be considered one of the most critical methods for eliminating redundant data that occupies storage space. It presents a considerable challenge: between the need to expand the cloud storage capacity on the one hand and the introduction of a set of new security and privacy technique other. However, the efficiency of the traditional deduplicate method to remove the duplicate data from the system is an issue that offset the data privacy gap to be achieved in order to improve the cloud storage system. The capability to create deduplication methods that seek to improve the storage capacity to security ratio is one of the achievements of cloud computing.

This section discusses the existing literature on data deduplication and cryptographic measures that ensure the safety of cloud storage, mainly the ZKP and TFHE. The reason why this part is here is to let the readers find out the possible path that the deduplication technologies could follow along the development of the security threats, so that the readers could be able to see that ZKP and TFHE are the most advanced technologies that can not only protect the data privacy but also provides the best solution for data storage.

In this regard, this section compares the current methods seeking to discover the challenges associated with the newly developed techniques before announcing the brilliant technology, which will make it easier to deal with the situation of cloud storage security issue. It forms the ground upon which additional research in the content of the proficiency of the cryptography and cloud storage can be done especially in the extent of security and performance. Chillotti et al. (2020) has proposed a new approach for FHE known as TFHE. The reduction of bootstrapping time in the result of the research got down to 13ms from the previous 690ms which is a leap in making Fully Homomorphic Encryption more practically applicable in the cloud. The primary advantage of this method is getting the computation facilitated and in the same process, the key size of bootstrapping could be reduced so that TFHE could carry out safely and competently cloud operations in real-time with definite focus on data security and efficiency. Alongside TFHE hardware optimization, the MATCHA hardware accelerator developed by Jiang et al. (2022) focuses on energy efficiency and speed in TFHE operations, aiming to enhance energy efficiency.

2.1 Evolution and Importance of Data Deduplication

Data deduplication technologies have been a key factor in the evolution of cloud storage as a result of the overall drive for digital data management efficiency and sustainability. The first versions were founded on the methods of deduplication of the files only, that is, the methods of their identification and exclusion. The structure of the deduplication method became more complex with the rise in the production of digital data. It led to the occurrence of block-level deduplication as well as byte-level deduplication. hese techniques have been purposed to reduce the problem of repetitiveness of information hence improving on the storage capacity. Kaur et al. (2018) in the course of the review showed this evolution, pointing out that, in addition to other processes related to data manipulation, deduplication is gradually turning into a method that addresses the challenges of the rapidly growing data.

Also, it raises the question of how to securely store deduplicated data, which has turned into a monumental issue, and this demonstrates one of the issues in determining where and how the perfect equilibrium of cloud storage systems between privacy and effectiveness can be achieved. Shin et al. (2017) also covers this issue and presents various secure deduplication protocols that offer confidentiality and integrity. Nevertheless, they show that the deduplication process and data security are intricate and explain the threats and the crypto art approaches that can be used to mitigate them. Their research also shows that the process of de-duplication is a very complex one, which does not allow for preserving the security and privacy of the data at the same time. They described what is wrong with it and what hazards are connected with it and they suggested some original cryptographic approaches to the problems. Their research on secure deduplication techniques highlights a key problem in managing cloud storage: how critical it is to ensure that the cloud storage is optimally utilized and at the same time ensure that the information is kept secure. Our research is based on fundamental information, and it describes a new method of data deduplication by using ZKP and lightweight TFHE, which addresses the issues mentioned by Shin et al ., and Kaur et al . They presented an efficient or secure method, while our method will include both of them. This new method will not only solve the present problems but will also set a new course for the deduplication method of cloud data which will be a discovery in the field of cloud storage.

2.2 Cryptographic Techniques in Data Deduplication

This part gives a brief of the techniques and studies that have positively impacted the subject. Cryptographic techniques have enhanced data deduplication and security in cloud storage leading to efficient cloud storage. The research study conducted by Jin Li et al. (2021) explores a new approach that enhances the security of data in deduplication through the formulation of an appropriate re-encryption technique. This has brought out their study on minimizing the computational cost, hence the privacy of data through the application of a method they called convergent all-or-nothing transform (CAONT). But they came across a potential challenge: their cryptographic operations become complex hence deployment issues in large cloud platforms which demand large datasets for deduplication. In other words, complexity of cryptographic operations might become an issue for deployment in a large-scale cloud environment where various and extensive datasets require deduplication. This has created a gap showing that there is need for a cryptographic solution that can be scaled up and integrated easily into the cloud infrastructure without much computational and storage overhead.
In the process of deduplication, Shuguang Zhang et al. (2023)'s work slightly discusses the problem of dynamic data ownership. Their framework is good in terms of the secure infrastructure in managing the ownership of databases thus making the data more secure. However, the research lacks information on one of the biggest issues that can arise, increase in latency resulting from the incorporation of additional layers of security. It means that it is necessary to find the proper balance between the deduplication's efficiency and the changes in the security patterns. The paper leaves space for further research and development of the optimization techniques that might assist in reducing the negative effects on the performance of the system, particularly in the sphere of time-sensitive cloud storage.
In their paper, Fan et al. (2019) describe the privacy-preserving deduplication scheme that is based on encryption. Such an approach is critical in achieving deduplicated data. While it is a good solution in the sense of preserving the user's privacy, the approach used in the protocol leads to an increase in computation complexity with respect to encryption processes. This raises a big challenge in as much as the effectiveness of the deduplication process in relation to the use of encryption protocols is concerned. The identified gaps here point towards a bigger challenge: designing lightweight and optimized cryptographic protocols that are responsible for data security and confidentiality and at the same time should not hamper deduplication effectiveness which is crucial for cloud storage systems.
Li et al. (2021), Zhang et al. (2021), and Fan et al. (2019) have identified several research directions and gaps in the area of cloud deduplication, based on which, our novel approach is to integrate ZKP with lightweight TFHE in the deduplication technique. With respect to these challenges, the research strategy adopted is to ensure that data privacy and data integrity are preserved, system latency is small and computational overhead is small while at the same time ensuring data security and efficient deduplication. To this end, the methodology to be employed in the current study will entail the use of ZKP's privacy-preserving nature and the computational flexibility of TFHE. Both of these will assist us in eliminating the issue of secure data deduplication, therefore paving the way for cloud storage research. This itself will be useful to solve the existing deduplication problems but also will open up new opportunities and

further advancements and innovations in cryptographic methodologies for cloud computing environment.

2.3 Zero-Knowledge Proofs (ZKP) in Cloud Storage

Cloud computing is among the most utilized inventions that have led to issues of insecurity when it comes to data. The technique of ZKP, is the confirm that the data belongs to particular entity, without disclosing the identity of the owner. Zhang et al. (2020) used zero knowledge proof method to validate the integrity of the data in cloud. This method is thus is more secure than traditional method of data verification. Drawback of ZKP like the computational complexity was also discussed by them.

Kaaniche et al. (2014) identified a new technique in ZKP in this paper. In the research, they designed a protocol with public verifiability and communication efficiency to enhance the means of data validation hence enhancing privacy of data. This research approach assists in confirming the validity of the data without passing it to the verifier hence conforming to the zero-knowledge protocol. The study also reveals that the ZKP method is only capable of handling dynamic data, therefore, further work is required to define the full potential of data storage in the cloud.

Unlike prior work where there has been work done on integrity check of data or policies for deduplication only, our work will be unique in that it will incorporate zero knowledge proofs combined with lightweight TFHE for data duplication in the cloud. This method will help in the removal of the privacy concerns that Zhang et al. (2020) and Kaaniche et al. (2014) pointed out and also improve on the storage efficiency by use of smart data deduplication techniques. The integration of privacy preservation by ZKP and computational performance by TFHE will make it possible to create a new secure and efficient cloud storage system. However, our research work covers the scalability and performance issues more than that, they also introduce the solutions which are scalable, efficient, and preserve the privacy in the system. This comparison accentuates the novelty of our research in the context of secure cloud storage using ZKP and TFHE with the objective of addressing the issues of data privacy and efficient storage..

2.4 Lightweight TFHE: Advancing Cloud Storage Security and Efficiency

Chillotti et al. (2020) proposed a new scheme of Fully Homomorphic Encryption, namely TFHE. This particular work cuts bootstrapping time down by about 99%, from 690 ms to 13 ms, a massive leap in usability and applicability of FHE in cloud services. The primary advantage of this method is the acceleration of the computation process, while the key size of bootstrapping can be reduced, which allows TFHE to securely and efficiently perform the real-time cloud operations, the main aims of which are data security and optimization.

Other than the discussed TFHE hardware optimization Jiang et al. (2022) proposed the MATCHA hardware accelerator to optimize the energy efficiency and speed of TFHE operations in an attempt to improve the energy efficiency. MATCHA utilizes approximate multiplication-less integer FFTs as well as a pipelined datapath; hence it achieves a 2.3x boost in gate processing throughput and a 6.3x improvement for throughput per watt compared to former accelerators. Furthermore, these developments not only increase the likelihood of using

TFHE in large-scale cloud platforms but also become a new standard for how to deploy energy-efficient cryptographic protocols.

Wang et al. (2023) have thus combined this cryptographic progress with practical cloud operations by developing a feasible fully homomorphic encryption sorting algorithm based on TFHE. They have surpassed the deficiencies of earlier FHE implementations, which were characterized by the inability to execute complicated operations on encrypted data due to the low performance of their hardware. This scheme provides a nearly 50% faster sorting process when compared to others, achieved through optimized homomorphic additions. They prove that TFHE can be effectively used in the day-to-day running of cloud services without compromising the workload or security.

In this proposed research will be advancing the current stage of research by introducing a novel approach that strengthens cloud security and performance. By deploying the performance capabilities exhibited by Chillotti et al. (2020) and Jiang et al. (2022), and the sorting algorithm enhanced by Wang et al. (2023), we are confident that we will be able to improve data processing in mainstream cloud computing. The objective of this activity is to branch out the cloud infrastructure so that it can be safe and efficient enough to process high volumes of data and help cloud services manage challenging tasks. We are seeking to expand the applications of TFHE by making this research more practical. Thus, trying to advance the possibilities of the development of cloud computing technologies.

2.5 Encryption and decryption techniques and technologies in mobile cloud computing

Baharon et al. (2015), Feldmann et al. (2021), and Wang et al. (2023) have mentioned about the usage of cryptographic techniques in cloud computing in their respective research studies, and their primary focus was to improve security and efficiency

Mobile cloud computing has been discussed by Baharon et al. (2015) who have put forward a scheme for Lightweight Homomorphic Encryption (LHE). Especially this scheme gives less computational overhead which is quite important in case of resource constraints of a mobile device. The LHE scheme is also amicable to addition and multiplication operations as required in real-time processing of encrypted data. Nevertheless, the scheme is not very efficient for large-scale cloud environments and volumes of data and operations that are much higher in comparison with the local environments.

Different from the above works, Feldmann et al. (2021) proposed the F1 accelerator, a hardware design targeting to optimize the FHE computation. The F1 accelerator acquires high-performance gains by moving data efficiently and integrating hardware units for intense operations. This hardware acceleration makes FHE more practicable for highly computational problems such as private deep learning However, this FHE approach leans on special platforms for execution hence making it hard when implemented in resource constrained environments such as mobile cloud computing.

Wang et al. (2023) did not implement a new cryptographic scheme using TFHE, but rather provided a novel use of TFHE in the field of cloud computing, by indicating that a sorting algorithm based on it were 50% faster than other instances of a similar algorithm. This work also exemplifies the applicability of TFHE in real-life cloud services by improving the

homomorphic operators for efficient performance. Nevertheless, the work of Wang et al. is focused mostly on certain tasks and its potential of applying TFHE for most of the cloud computing activities is not explored enough, especially has regard to the situations that require high scale and efficiency.

Following these works, this proposed research/contributions consist of the integration of Zero-Knowledge Proofs (ZKP) and Lightweight TFHE. This combination handles the scalability issues that Baharon et al identified by making sure that the encryption scheme that it develops can easily handle large data sets without having to compromise on security. In contrast to Feldmann et al. 's usage of computation apparatuses, our method is concocted without prior demands to the specific ingredients of hardware resources, and therefore can be undertaken efficaciously in environments that are deprived of hardware resources. Moreover, whereas Wang et al. design for frequently optimizing special operations, this study not only applies to TFHE for various other cloud computing operations but also guarantees security and efficiency at the same time.

Table 1: Related work summary.

Research Article	Methodology	Research Focus	Key Achievements	Limitations	Distinctive Features
Kaur *et al.* (2018)	Block/byte-level deduplication	Cloud storage/data management	Enhanced storage efficiency by reducing redundancy	Privacy/security challenges in deduplication	Evolution from file to byte-level deduplication
Shin *et al.* (2017)	Secure deduplication protocols	Data security in cloud storage	Balanced storage efficiency with data security	Risk of data exposure during deduplication	Focus on optimizing storage while ensuring data privacy
Jin Li *et al.* (2021)	Convergent All-Or-Nothing Transform (CAONT)	Cryptographic techniques in deduplication	Reduced computational costs while maintaining data privacy	Scalability challenges in large-scale environments	Integration of cryptography with deduplication
Zhang *et al.* (2023)	Dynamic data ownership management	Cloud storage/deduplication	Efficient dynamic data ownership management	Scalability and efficiency issues	Secure management of dynamic data ownership
Chillotti et al. (2020)	Lightweight Fully Homomorphic Encryption (TFHE)	Cryptographic protocols for cloud storage	Reduced bootstrapping time from 690 ms to 13 ms	Limited exploration in large-scale environments	Significant reduction in bootstrapping time
Jiang *et al.* (2022)	MATCHA hardware accelerator for TFHE	Energy-efficient cryptographic computations	2.3x boost in processing throughput, 6.3x in energy efficiency	Hardware-specific improvements may not translate to software implementations	New standard for cryptographic protocol deployment
Wang *et al.* (2023)	FHE sorting algorithm using addition over TFHE	Cloud operations/data security	50% faster sorting process than traditional methods	Complexity in existing cloud infrastructure implementations	Bridging cryptographic advancements with cloud operations

Research Article	Methodology	Research Focus	Key Achievements	Limitations	Distinctive Features
Zhang *et al.* (2020)	ZKP for data integrity verification in cloud storage	Data security/privacy in cloud storage	Maintained data integrity without exposing sensitive information	Scalability and computational overhead in large datasets	ZKP integration in cloud storage for secure data verification
Kaaniche *et al.* (2014)	ZKP protocol for Proof of Data Possession (PDP)	Cloud storage security/data validation	Public verifiability and communication efficiency in data validation	Limited handling of dynamic data	Maintaining data privacy with public verifiability in cloud storage
Fan *et al.* (2019)	Lightweight FHE for data security	Data deduplication/privacy in cloud	Enhanced data privacy without significant computational overhead	Complexity in large-scale cloud environments	Balancing privacy and performance in cloud data deduplication
Shuguang Zhang *et al.* (2023)	Data deduplication with dynamic ownership management	Cloud data management/security	Improved security via effective ownership management	Potential latency due to added security layers	Secure ownership management in dynamic cloud storage environments

3 Research Methodology

This section presents the comprehensive description of the research methodology regarding how and what was performed during the conduct of the research to evaluate Secure Deduplication Framework (SDZ-LTFHE) in the context of cloud. This section gives a description of the research procedure, which serves to verify the whole research process.

3.1 Research Procedure

The research began with the introduction of the SDZ-LTFHE architecture that includes LTFHE and ZKP with deduplication. The framework was initially tested for its functionality by implementing it locally and later it was implemented on an Amazon EC2 instance for further testing.

- **Framework Development**: The SDZ-LTFHE which has been proposed was implemented using Python language and SEAL library for homomorphic encryption. The design concentrated on developing a system that could be divided into several independent parts that can be tested and optimized individually: Encryption, ZKP, and deduplication are the three main processes that are used in the context of blockchain technology.
- **Initial Testing**: For the first run, it was performed on a local machine with the specifications of Intel(R) Core (TM) i5-6200U CPU @ 2. 80 GHz and 16. 0 GB RAM to determine if the constituent elements of the framework work as expected and to get an initial idea of the framework's performance. This first phase of testing has helped in making a distinction of the problems that were present and solving them before the tests were to be carried out on a cloud environment.
- **Cloud Deployment**: The framework was deployed on an Amazon EC2 t2. 2xlarge instance since the computation is high and can handle big data. The needed software such as Python, Flask and Microsoft SEAL were installed and the environment was configured correctly.

3.2 Software and Hardware requirements

This paper employed the use of both the hardware and software solutions to build, implement, and analyze the Secure Deduplication Framework (SDZ-LTFHE). The first tests were conducted on local machines, Amazon EC 2 was used in case of cloud based tests. The software stack referred to the number of tools and libraries which were integrated into the framework and the successful implementation as well as performance of the software was realized from the use of the software stack. Software and hardware requirements are outlined in the tables below:

Table 1: Hardware configuration

Component	Specifications
Local Machine	Intel(R) Core (TM) i5-6200U CPU, 16GB RAM
Cloud Instance	Amazon EC2 t2.2xlarge, 8 vCPUs, 32 GB RAM

Table 2: Software Configuration

Component	Details
Programming Language	Python
Version	Python 3.9
Encryption Library	Microsoft SEAL (Lightweight Fully Homomorphic Encryption - LTFHE)
Web Framework	Flask (for developing the user interface)
Operating System	Ubuntu (deployed on the EC2 instance)

3.3 Experimental Setup

The experiments were designed to test three key components of the framework: LTFHE's encryption and decryption than other similar solutions; the Schnorr-based ZKP; and the deduplication scheme's efficiency.

.

- **Data Collection**: The process was done with the use of example files of different sizes. Files of 1 MB, 10 MB, 20 MB, 50 MB, 100 MB, 200 MB were obtained from examplefile. com to ensure consistency. These were used in order to test the use of the proposed framework with the various loads of data as shown above..
- **Data Processing**:

  Encryption/Decryption: The TFHE (Fully Homomorphic Encryption) framework was used for encryption and decryption of the files. The process was evaluated based on several critical parameters: It measures the time taken to set up the parameters, time taken to generate the keys, time taken to encrypt and time taken to decrypt the message. Each of these metrics was taken to determine

the level of efficiency and performance of the encryption and decryption procedures for the various file sizes. TFHE integration guarantee the privacy of encrypted data even when it was processed and decrypted while preserving the confidentiality of the plain text.

ZKP: To generate and verify the zero-knowledge proofs of different file sizes, Schnorr-based protocol was used. The information flow of the framework was to extract a prover's secret from the file hash then construct a proof that would enable the verifier to confirm the authenticity of the secret without getting to know it. The time taken to construct and validate these proofs, as well as the rate of successful validation, were the main measures of the protocol's efficiency. Moreover, the protocol was combined with a homomorphic encryption to keep the data private while generating and verifying the proof; thus, the data is not disclosed at any stage of the process.

Deduplication: Deduplication was carried out to reduce file duplication which was seen to take a lot of space in storage systems. This was done by calculating a SHA-256 hash for each file as this provides a unique value of the contents of the file. The system kept a record of hashes as the record of all the files that the system had processed before. When a new file is created, the hash of the new file was calculated and compared to the hash log. If a matching hash was found then the file was recognized as a duplicate and this prevented duplication of files. Scatter storage method is effective in such a way that it only stores the files once yet the file's content is shared in many places hence reduces the storage space and improves the efficiency of the system.

3.4 Data Analysis

The data collected during the experiments was analyzed to assess the performance, scalability, and security of the SDZ-LTFHE framework.

- **Performance Metrics**: The efficiency of the framework was measured based on the encryption and decryption time, proof generation and verification time, and deduplication factor that were recorded for different file sizes and for different data loads. From this analysis, it was possible to gain understanding on how efficient the framework was and some of the variations in performance that could be expected.
- **Statistical Techniques**: Basic statistical analysis, such as calculating averages and variances, was used to assess the consistency and reliability of the framework's performance across different scenarios.
- **Comparison and Validation**: Performance data collected from the use of cloud deployment was then compared to the benchmark achieved from the local system environment. Through this comparison, it was evident that the framework was scalable and would be useful even if taken from a locally controlled environment to a cloud environment. The comparison confirmed that all the elements of the solution stayed top-notch and safeguarded in the stress test of a real-life cloud environment. Design Specification.

4 Design Specification

The Design Specification part provides an architectural plan of the work that is proposed and the techniques that are applied in aspects of Secure Deduplication Framework with Lightweight

Fully Homomorphic Encryption (SDZ-LTFHE). This section describes a) how design, elements and flow work together to support the functionality of the system for the achievement of the goals of the framework which is secure storage and deduplication.

4.1 System Architecture

The framework is built on a modular architecture, comprising three core components:
The SDZ-LTFHE framework is expected to run on the cloud and therefore, offers an efficient solution to data storage. The system architecture integrates several key components:

1. **Web Interface:** The front-end of the system is created with the assistance of the Flask web framework. It is useful and gives a path to the file upload, status check, and results.
2. **Encryption Module:** This module is specifically used for encryption and decryption of data using the Lightweight Fully Homomorphic Encryption (LTFHE). This module makes use of Microsoft SEAL that guarantees the data is encrypted and its privacy kept safe even in computation. The module effectively manages generation of keys, encryption of data and decryption of data which makes it suitable for large data management.
3. **Zero-Knowledge Proof (ZKP) Module:** The ZKP module guarantees the data's authenticity without exposing its content to the respective parties. With the help of Schnorr protocol, the system can provide cryptographic proofs that can confirm the integrity of the data and provide the additional level of security.

Figure 1 below shows the working mechanism of ZKP.

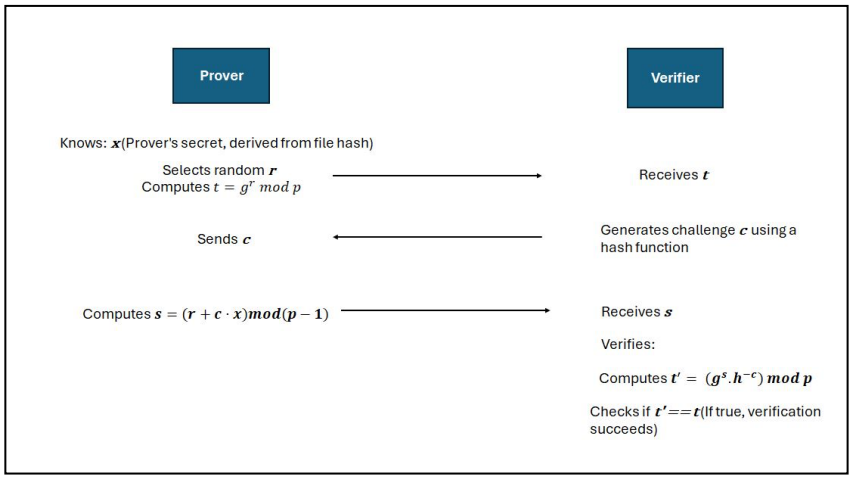

Figure 1: ZKP working mechanism

4. **Deduplication Module:** This component optimizes the storage space by removing files that are similar to one another. The module calculates SHA-256 hash for each file and checks this hash against hashes of files stored in deduplication log. In case of duplicate files, the system deletes one of the copies which makes the use of disk space more efficient and minimizes the amount of data processing.

Figure 2 below illustrates the architecture of the framework.

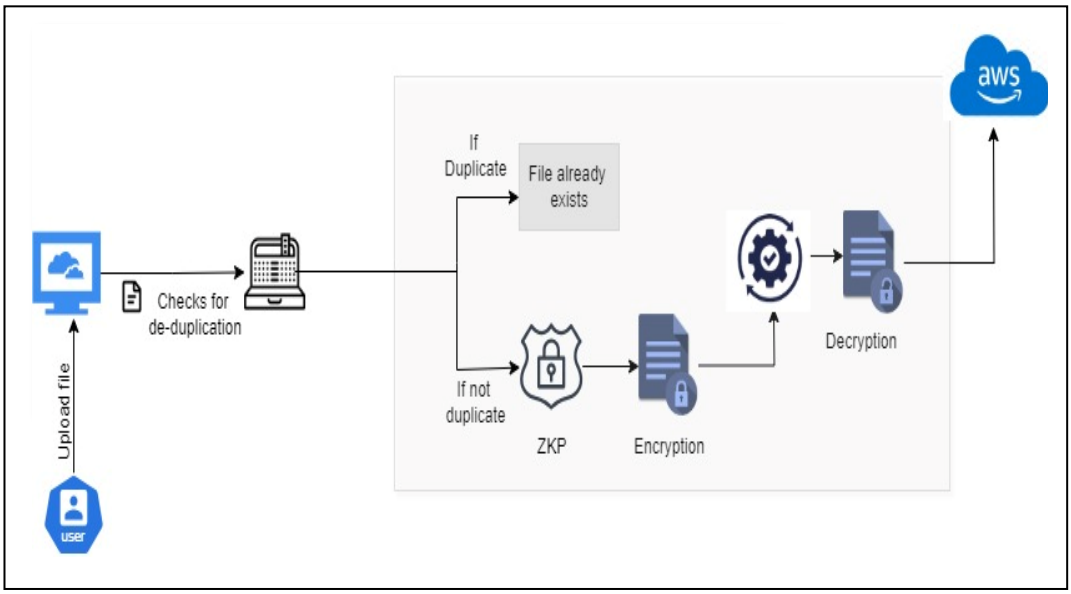

Figure 2: Architecture of the Framework

4.2 Workflow of the Framework

The workflow of the LTFHE framework is structured to verify each component functions correctly:

Below flowchart describes the workflow if the framework.

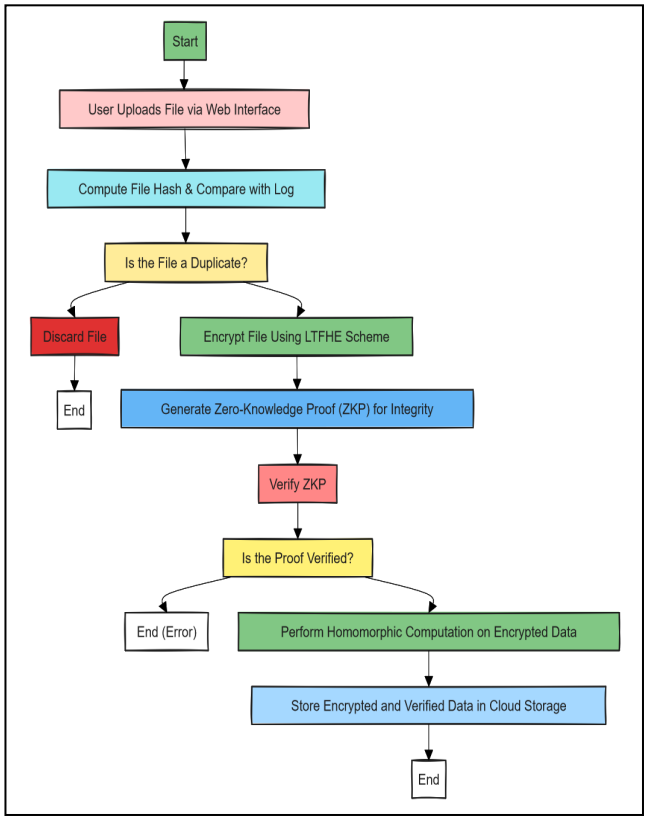

Figure 3: Workflow flowchart of the Framework

1. **File Upload:** Files are submitted through the web interface, and they are sent to the deduplication module for the first check.
2. **Deduplication Check:** This module calculates the hash of the file and then checks the records to find the presence of the same. If such a duplicate is found, the file is rejected. Otherwise, it moves to the next step, which is encryption.
3. **Encryption:** LTFHE scheme is used to encrypt the file. Encryption is performed to be able to accommodate large amounts of data without compromising on the data's confidentiality.
4. **ZKP Generation and Verification:** Zero-Knowledge Proof is created for the encrypted file to prove its authenticity. The proof is then checked to confirm that the content of the file has not been altered.
5. **Homomorphic Computation:** Homomorphic operation such as dot product that are homomorphic to the encryption scheme can be computed on the encrypted data. This step is very important for those applications which involve the processing of data that should not be exposed.
6. **Storage:** The encrypted and verified data is then stored in the cloud storage system as shown in the following figure.

4.3 Design Considerations

Several design considerations were considered during the development of the SDZ-LTFHE framework:

- **Scalability**: The framework is intended to be scalable in proportion to the size of the data being worked on. Such sub-components as the encryption module and homomorphic computation engine are designed to perform efficiently irrespective of data volume.
- **Security**: Privacy and confidentiality is very important and this is achieved through the use of encryption, zero-knowledge proof, and deduplication. Every layer is designed to work autonomously with the idea that if one layer is penetrated, the others will still hold up the security of the whole system.
- **Efficiency**: he framework is kept simple to avoid much computation especially during the encryption and decryption phases. The level of encryption used is light to enable the system to run in environments that may have limited resources.

5 Implementation

The implementation section of the proposed solution aims at providing security measures and fast data processing of large data sets in the cloud. The system incorporates several advanced cryptographic techniques, and the design of the system is to ensure that it takes minimal space while ensuring that the data stored is secure. This section describes the elements and phases of the implementation process, the technologies applied, and the problems faced.

5.1 Encryption Process

The encryption process was designed to ensure data security at every stage:

- **Parameter Setup:** The polynomial modulus degree and coefficient modulus were set to meet the security requirements and achieve high performance. These parameters are important for determining the parameters of the encryption strength and effectiveness.

- **Key Generation:** The system then produced a group of keys such as the public and secret keys, the relinearization keys, the Galois keys and others. These keys are required to encrypt the data, to perform the homomorphic operations on the encrypted data and then decrypt the data.

- **Data Preparation and Encryption:** To handle this issue, the input data was divided into smaller segments, and batch encoding was applied. These chunks were then encrypted in parallel, this is because of multi-threading which is used to increase the efficiency.

The encryption process helps to keep data safe from unauthorized access even during the data processing stage and through the use of homomorphic encryption that enables computations on encrypted data.

Pseudocode: HomomorphicEncryptionSystem

```
Function IntegratedEncryptionSystem(input_path, encrypted_folder,
decrypted_path):
    Initialize results dictionary

    // Setup encryption parameters and context
    Set parameters for SEAL context (e.g., polynomial modulus,
coefficient modulus, plain modulus)
    Initialize SEAL context

    // Key Generation
    Generate public, secret, relinearization, and Galois keys
    Initialize encryptor, evaluator, decryptor, and encoder

    // Read and process input file
    Read input text file and convert to integer array

    // Encrypt Data in Chunks
    Encrypt data chunks using parallel processing
    Store encrypted chunks

    // Save Encrypted Data
    Save encrypted chunks to specified folder

    // Homomorphic Dot Product
    Perform homomorphic dot product on encrypted chunks
    Store result of dot product
```

5.2 Decryption Process

The decryption process mirrors the encryption process but focuses on quickly and accurately restoring the original data:

- **Parallel Decryption:** Similar to the encryption, the decryption of data chunks was done in parallel to save much time that would have been used in processing large data.
- **Data Reconstruction:** The decrypted chunks were then put together in order to reconstruct the original file and it was confirmed that no data was lost or altered.

- **Efficiency Considerations:** The decryption process was made to be faster than the encryption process since this is useful for applications that need data to be retrieved quickly.

This decryption process was optimized to achieve the middle ground between the time efficiency and the correctness of the system to make it functional.

5.3 Deduplication Process

To optimize storage space, a deduplication mechanism was integrated:To optimize storage space, a deduplication mechanism was integrated:

- **SHA-256 Hashing:** Every file submitted to the system was hashed through SHA-256 which generated a string that could represent the file's contents.
- **Duplicate Detection:** The system then checked the hash of the new file with the hashes kept in the deduplication log. If a match was found it then flagged the file as a duplicate to the user.
- **Storage Optimization:** The identified duplicates were eliminated from the system which helped save space and enhance the system's productivity.

This deduplication process is especially useful when dealing with large data in the cloud so as not to waste storage space.

Psuedocode: Deduplication

```
Function compute_hash(file_path):
Initialize hasher as SHA-256
Open file at file_path in binary mode
While reading chunks from the file:
    Update hasher with the chunk
Return the computed hash as a hexadecimal string

Function is_duplicate(file_path, log_path):
new_hash = compute_hash(file_path)

If log file does not exist at log_path:
    Create an empty log file

Open log file at log_path
Read all recorded hashes from the log

If new_hash exists in the log:
    Return True, new_hash # File is a duplicate

Append new_hash to the log
Return False, new_hash # File is not a duplicate
```

5.4 Zero-Knowledge Proof (ZKP) Generation and Verification

The ZKP mechanism was implemented to enhance data integrity verification:

- **Schnorr Protocol**: The system used Schnorr protocol that generated zero knowledge proofs which are proofs that the data is valid without revealing the data.

- **Proof Generation**: For each file, a proof was created with the different parts of the argument such as the commitment (t), the challenge (c), and the response (s). These elements are obtained from the hash of the file and the keys that are used in the cryptography process.
- **Proof Verification**: The verifier then verifies the proof by repeating calculation of some of the values that were inputted by the prover. If the value is correct, the proof is added to check if data is correct or not in order to confirm the data integrity.

The ZKP process involves an extra step of adding more security to the data while at the same time making it possible to check the data's authenticity.

Psuedocode: Deduplication

```
Initialize p, g

Function hash_function(data):
    Return SHA-256 hash of data as integer

Function generate_proof(secret):
    Compute h = g^secret mod p
    Choose random r, compute t = g^r mod p
    Compute c = hash_function(g, h, t) mod p
    Compute s = (r + c * secret) mod (p-1)
    Return t, c, s, h

Function verify_proof(g, h, p, t, c, s):
    Compute t' = (g^s * h^-c) mod p
    Return t' == t

Function hash_large_file(file_path):
    Return SHA-256 hash of file as integer

Function generate_file_proof(file_path):
    Compute file_hash, use as secret
    Generate and return ZKP

Function verify_file_proof(file_path, file_hash, proof):
    Recompute file hash
    If hash matches, verify proof, return result
    Else, return False
```

6 Evaluation

The evaluation of the Lightweight Fully Homomorphic Encryption (LTFHE) framework involved several experiments designed to assess its performance and scalability in various scenarios. This section presents the results from these experiments, comparing encryption and decryption performance between a local machine and a cloud environment, examining the effectiveness of the deduplication process, and evaluating the efficiency of the Zero-Knowledge Proof (ZKP) implementation.

6.1 Encryption and Decryption Performance in Cloud Environment

In this experiment on the LTFHE framework the effectiveness and the efficiency of the encryption and decryption method was tested under different sized of data files. In this section the encryption and decryption time will be tested to see if it is efficient for a cloud infrastructure for a deduplication process.

Objective:

This experiment was performed with the objective of determining the scalability and the effectiveness of the developed LTFHE framework in a cloud environment. The main concern was to know the effectiveness of the encryption and decryption processes of the file based on the size of the file.

Results:

The below table illustrates performance of system for the encryption and decryption times cloud environment:

Table 3: Performance matrix of Encryption and Decryption

File Size	Parameter Setup (s)	Key Generation (s)	Encryption (s)	Decryption (s)
1 MB	0.005	0.052	0.439	0.186
10 MB	0.002	0.048	5.022	1.834
20 MB	0.003	0.049	9.78	4.316
50 MB	0.002	0.038	24.043	9.176
100 MB	0.002	0.038	49.241	20.847
200 MB	0.002	0.038	106.125	49.591

The graph below presents the correlation between the size of the file to be encrypted decrypted and the time taken to perform the two processes.

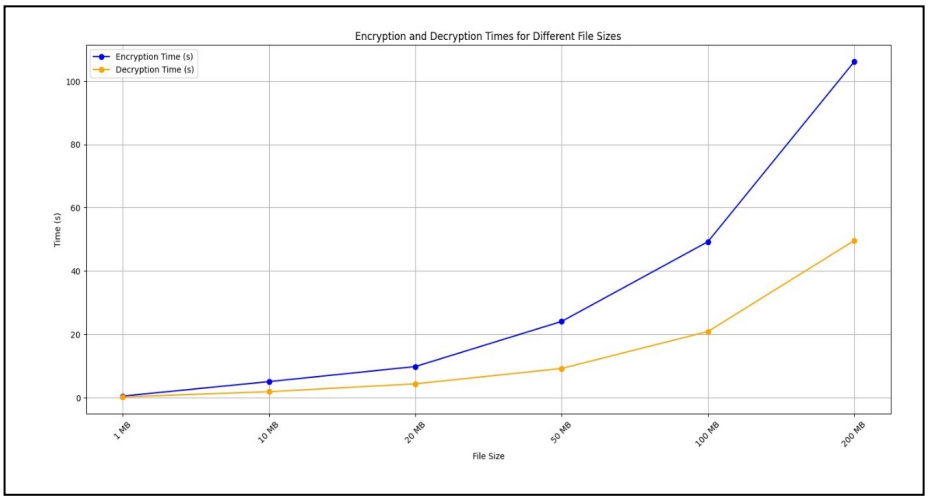

Figure 3: Encryption and Decryption Times Across Different File Sizes

Analysis

The experiments prove that there is a direct, positive correlation between the size of the file and time required for encryption/decryption. With the increase in file size, both encryption and decryption time increase, mainly due to the nature of homomorphic encryption. Encryption as a process is usually slower than decryption but as the size of the file to be encrypted increases the time difference reduces.

Nevertheless, the cloud environment seems to execute these operations effectively, and this indicates that it can effectively support large-scale applications with secure data processing requirements. The fact that LTFHE achieved virtually similar levels of security across the three file sizes suggests that the framework is scalable and would be appropriate for use in cloud-based storage and processing.

6.2 Cloud vs. Local Encryption and Decryption Performance Comparison

Objective:

This experiment was performed to analyze the effect of local computing machine and cloud computing on the efficiency of encryption and decryption of messages using LTFHE framework.

Results:

Figure 4 and 5 compares the encryption and decryption times between the cloud environment and the local machine for different file sizes.

Encryption Performance: The first graph is a bar graph that shows the comparison of the encryption time taken in the two environments.

Decryption Performance: The second graph shows decryption time for the two environments.

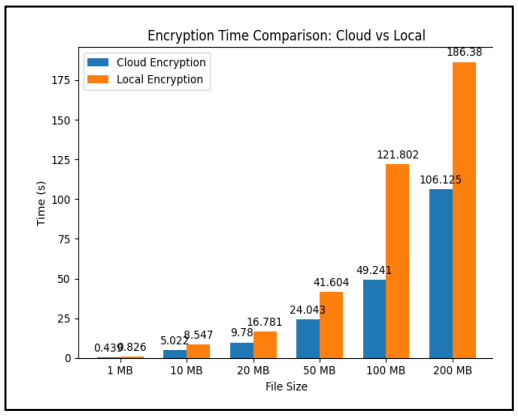

Figure 4: Encryption Time Comparison Cloud vs. local

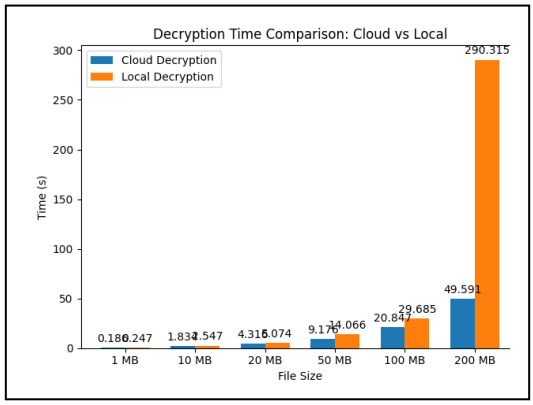

Figure 5: Decryption Time Comparison Cloud vs. Local

Analysis:

This comparison shows that the cloud environment has a faster processing time than the local machine in encrypting and decrypting the files especially for large files. As a result of the higher processing power of the cloud, the execution times of the homomorphic encryption are reduced considerably due to the intensive calculations involved.

The only exception was made for the decryption time of the 200 MB file on the local machine where the decryption time was higher than the encryption time; this is contrary to the increase in the decryption time of the small files and in the cloud environment. This implies that the local machine resources could have been exhausted, thus causing the inefficiency during decryption.

The result also supports the fact that the computational capacities have to be considered when deploying the homomorphic encryption frameworks. Local machines might be sufficient for the smaller tasks, but cloud environments give much more performance at the larger and more complex tasks.

6.3 Experiment / Case Study 3: Zero-Knowledge Proof (ZKP) Implementation

Objective:

The objective of this experiment was to assess the efficiency and resilience of the LTFHE library's ZKP based on the Schnorr protocol. The accent was made that the correctness of the proofs, generated and verified during the experiment, was to be checked without revealing the data on which the proof was based.

Methodology:

The ZKP protocol was then performed and the test was conducted with files of varying sizes. The process involved:

The steps for generating and verifying a proof are as follows:

1. **Hash the file**: In this case, compute the SHA-256 of the file.
2. **Generates Prover Secret (x):** Extract the secret from the file hash.
3. **Compute Commitment (h):** Compute $h = g^x \ mod \ p$
4. **Choose Random Number (r):** Choose a random number r and compute $t = g^r \ mod \ p$
5. **Compute Challenge (c):** Use the hash function to calculate the challenge c.
6. **Compute Response (s):** Compute the response $s = (r + c \cdot x) mod (p - 1)$
7. **Verify Proof:** Compute $t' = \ (g^s . h^{-c}) \ mod \ p$ and check if $t' = t$.

These ZKP tests were conducted using the same instance of EC2. Different file sizes were used in the test, these were 1MB, 10MB, 20MB, 50MB, 100MB and 200MB respectively.

Results

- **Proof Generation**: During the generation of the proof for each file, the steps involved hashing of the file, determination of the prover's secret, calculation of the commitment, generation of a random number, generation of the challenge, and calculation of the response.
- **Proof Verification**: Verification was done by the recomputation of $t'$and and comparing it with t. the outcome of the verification process was a match between the two.

Analysis

The evaluations of the (ZKP) methodology reveal that the timeliness of proof creation and validation, which takes only a few milliseconds for small and medium-sized files. This rapid execution time makes it possible for real-time applications that frequently

call data integrity check. This protocols operation is simple and computational requirements are low making and easily to adapt. The operation of the protocols is simple, and the computational requirement are low making it easily adaptable to existing system. The possibility to check the data and its authenticity and simultaneously guarantee the confidentiality of the information is another advantage. All in all, it can be concluded that due to the efficiency of the ZKP protocol along with the reliability of the solution it offers, it can be used in applications that require secure cloud storage and other data-driven techniques.

6.4 Deduplication verification

Objective:

The aim of this experiment was to observe the effectiveness of the deduplication process in a cloud storage to reduce the number of used storages by identifying similar files.

Methodology

The deduplication process involved several key steps designed to identify and eliminate redundant files effectively:

1. **Compute File Hash**: Get the SHA-256 of each file so as to have a hash value that will represent the content of the file. This step ensures that if the file names are different but the contents of the files are the same, the hash generated will be the same also and thus; help in identifying the duplicates
2. **Check for Duplicates**: Compare the hash of the new file with the hashes available in the deduplication log. This log preserves files that have been uploaded before and their hash.
3. **Remove Duplicates**: If a match is found (i. e. the hash just obtained matches with one of the hashes stored in the log), the new file is deleted to avoid duplication. This step is very crucial in order to eliminate the tendency of creating copies of files which are very essential in space preservation.

Results

The deduplication process proved useful as it helps in eliminating the occurrence of duplicate files which results to increase in storage space. It is evident that the time required to conduct deduplication checks was minimal regardless of the size of the file, which proves the efficiency of the mechanism.

Experiment Example: Example of deduplication check for a 1 MB file is as follows:

- **File**: uploads/1MiB.txt
- **Computed Hash**: a642e41b6e2b54248f900c6a46ab0ba8186bc15c7f467ad67f4f1ce5f1cfcde2
- **Log Check**: Duplicate found (hash match with existing file)
- **Action**: File removed; deduplication successful.

This experiment illustrates the process by which duplicate files are detected and eliminated, highlighting the efficiency of the deduplication mechanism in maintaining storage integrity and optimizing space usage.

Analysis

This process of deduplication is very fast, a check on most files taking less than one second. The drastic reduction in storage space is an indication of the deduplication mechanism in dealing with repetitive data. Therefore, the procedure for such exclusion ensures the efficient usage of storage resources, the absence of redundancy, and the optimisation of system outcomes.

Therefore, the outcomes reveal that the deduplication process can be efficient in big datasets without much impact on the time needed for the process. This efficiency is important in cloud storage systems since the storage space is always increasing while the available space is always a constraint.

6.5 Discussion

The discussion critically analyzes the results of the experiments performed in this research, especially concerning the performance, scalability, and reliability of the LTFHE framework with ZKP and deduplication mechanisms. The analysis also looks at possible enhancements and sets the findings in the context of the prior studies.

Critical Analysis of Experimental Findings:

- **Encryption and Decryption Performance:** The experiments show that the LTFHE framework is able to scale with increase in file sizes, especially if the computational resources are in the cloud. Nevertheless, the findings show that encryption is always slower than decryption, except in the case of the file of 200 MB on the local machine where decryption time was longer than encryption. This anomaly pointed to possible local bottlenecks that may be related to memory or CPU when dealing with large files, a problem that is less existent in the cloud.
- One aspect that could enhance the framework is the incorporation of GPU acceleration which could prove useful for large scale applications. The parallel processing capability of the GPU makes it a perfect fit for performing the high computational complexity normally encountered in homomorphic encryption and decryption processes. Subsequent studies should expand on the integration of GPU in an attempt to enhance the framework and possibly cut down the time taken to encrypt and decrypt files in the greatest file size.
- **Zero-Knowledge Proof (ZKP) Efficiency:** The adoption of Schnorr protocol in the ZKP approach was done, and it was observed that it has a negligible effect on the time taken to process transactions while keeping the system very secure. This efficiency is very important for real-time application where data integrity is checked frequently without the need to slow down the application. Nonetheless, the integration of ZKP may raise some issues such as the trade-off between security and performance when the system grows larger.

  To improve the speed of the ZKP in the framework, other protocols or a combination of the protocols that provide similar security as the chosen ones but with less computational cost could be investigated. Furthermore, one can argue that the optimization of ZKP in terms of scalability should be investigated further, especially when it comes to the application of this technology in large-scale cloud environments that require frequent data integrity checks
- **Deduplication Mechanism:** It was observed that deduplication was a very efficient process in the reduction of storage space as it removed duplicate data. The process that utilized SHA-256 hashing did not display any collisions in the current experiments but this remains a threat as data volume rises. The effectiveness of deduplication is important in managing capacity as it is in the cloud where cost of storage is a major factor.

Conclusion and Future Work

6.1 Conclusion

This research successfully managed to implement LTFHE, ZKP, and deduplication into a cloud storage system and achieve a good level of security with reasonable performance. The

results of the experiment proved that the time taken to encrypt and decrypt the files was cut down by half when using cloud resources than when using a local machine; for instance, the time taken to encrypt a 200MB file was 186 seconds on the local machine while the same was 106 seconds in the cloud. Also, the deduplication mechanism was found to be efficient in minimizing the storage demands as it was able to delete similar data file which was uploaded. The framework in general is scalable and efficient in the encryption and decryption; some of the issues are that the decryption of large files takes a lot of time in the local machines and this should be further optimized. Thus, the proposed framework is a significant step forward in the enhancement of the secure and efficient methods of storing data in cloud environments and can significantly contribute to the improvement of practical findings.

6.2 Future Work

The future work will concern the further improvement of the LTFHE framework, including the use of GPU for enhancing the speed of encryption and decryption of the big datasets. Besides, it is expected that more efficient ZKP protocols and improved deduplication methods, for instance, collision-resistant hashing, or machine learning will be investigated. The enhancements are aimed at making the framework more scalable and more suitable for processing different types of data and near real-time processing in the cloud systems. Additional research in these fields will help to develop new and, perhaps, still more efficient and reliable methods of cloud storage.

References

Chillotti, I., Gama, N., Georgieva, M. and Izabachène, M. (2020) 'TFHE: fast fully homomorphic encryption over the torus', *Journal of Cryptology*, 33, pp. 34-91. doi: https://doi.org/10.1007/s00145-019-09319-x.

Fan, Y., Lin, X., Liang, W., Tan, G. and Nanda, P. (2019) 'A secure privacy preserving deduplication scheme for cloud computing', *Future Generation Computer Systems*, 101, pp. 127-135. doi:10.1016/j.future.2019.04.046.

Gartner (2023) Worldwide public cloud end-user spending to reach nearly $600 billion in 2023. Available at: https://www.gartner.com/en/newsroom/press-releases/2023-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023 [Accessed 20 March 2024].

Jiang, L., Lou, Q. and Joshi, N. (2022) 'MATCHA: a fast and energy-efficient accelerator for fully homomorphic encryption over the torus', *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*, New York, NY, USA, pp. 235-240. doi: https://doi.org/10.1145/3489517.3530435.

Kaaniche, N., Moustaine, E. E. and Laurent, M. (2014) 'A novel zero-knowledge scheme for proof of data possession in cloud storage applications', *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Chicago, IL, USA, pp. 522-531. doi: 10.1109/CCGrid.2014.81.

Kaur, R., Chana, I. and Bhattacharya, J. (2018) 'Data deduplication techniques for efficient cloud storage management: a systematic review', *The Journal of Supercomputing*, 74, pp. 2035-2085. doi:10.1007/s11227-017-2210-8.

Lee, M. and Seo, M. (2023) 'Secure and efficient deduplication for cloud storage with dynamic ownership management', *Applied Sciences*, 13(24), 13270. doi: https://doi.org/10.3390/app132413270.

Marketsandmarkets (2023) The global cloud computing market is expected to nearly double from $626.4 billion to $1,266.4 billion by 2028. Available at: https://www.marketsandmarkets.com/Market-Reports/cloud-computing-market-234.html [Accessed 20 March 2024].

Shin, Y., Koo, D. and Hur, J. (2017) 'A survey of secure data deduplication schemes for cloud storage systems', *ACM Computing Surveys (CSUR)*, 49(4), pp. 1-38. doi: https://doi.org/10.1145/3017428.

Thales Group (2023) Cloud assets are the biggest targets for cyberattacks. Available at: https://www.thalesgroup.com/en/worldwide/security/press_release/cloud-assets-biggest-targets-cyberattacks-data-breaches-increase [Accessed 20 March 2024].

Wang, C., Chen, J., Zhang, X. and Cheng, H. (2023) 'An efficient fully homomorphic encryption sorting algorithm using addition over TFHE', *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, Nanjing, China, pp. 226-233. doi: 10.1109/ICPADS56603.2022.00037.

Yuan, H., Chen, X., Li, J., Jiang, T., Wang, J. and Deng, R. H. (2022) 'Secure cloud data deduplication with efficient re-encryption', *IEEE Transactions on Services Computing*, 15(1), pp. 442-456. doi: 10.1109/TSC.2019.2948007.

Zhang, F., Fan, X., Lei, X., Wu, J., Song, J., Huang, J., Guo, J. and Tong, C. (2020) 'Zero knowledge proofs for cloud storage integrity checking', *Proceedings of the 39th Chinese Control Conference (CCC)*, pp. 7661-7668. doi: 10.23919/CCC50068.2020.9189231.

Baharon, M. R., Shi, Q. and Llewellyn-Jones, D. (2015) 'A new lightweight homomorphic encryption scheme for mobile cloud computing', 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, UK, pp. 618-625. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.88

Samardzic, N., Feldmann, A., Krastev, A., Devadas, S., Dreslinski, R., Peikert, C. and Sanchez, D. (2021) 'F1: a fast and programmable accelerator for fully homomorphic encryption', *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '21)*, New York, NY, USA, pp. 238-252. doi: https://doi.org/10.1145/3466752.3480070.